ENCICLOPEDIA PRACTICA DE LA

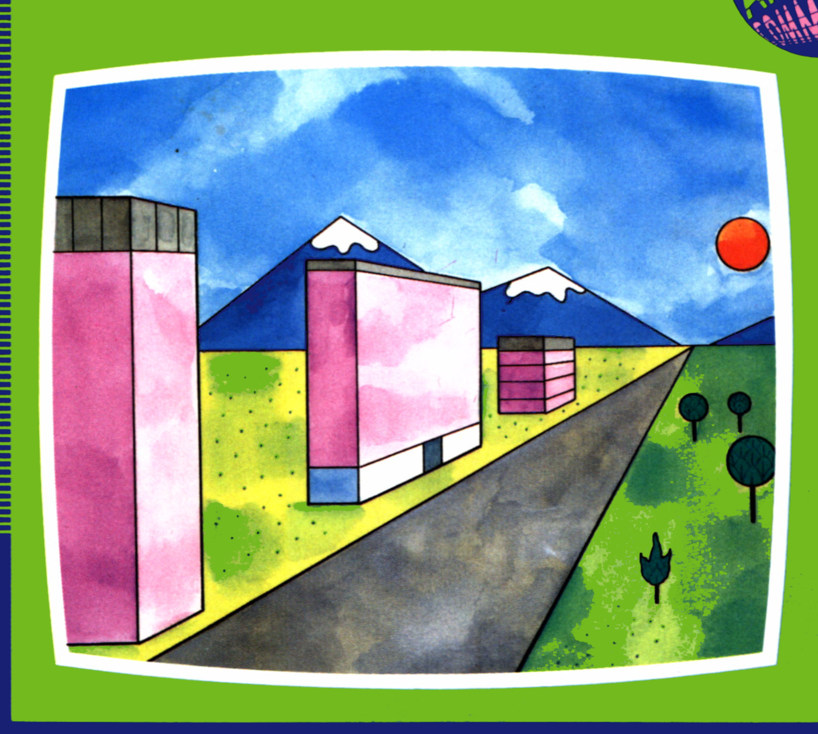
INFORMATICA APLICADA

2

Cómo dibujar y hacer gráficos con el ordenador

Aula de Informática

SINCLAIR
AMSTRAD
COMMODORE
IBM



EDICIONES SIGLO CULTURAL

ENCICLOPEDIA PRACTICA DE LA

INFORMATICA APLICADA

2

Cómo dibujar y hacer gráficos con el ordenador

EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática
JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño y maquetación:
BRAVO-LOFISH.

Dibujos:
JOSE OCHOA Y ANTONIO PERERA.

Tomo 2. **Cómo dibujar y hacer gráficos con el ordenador**

AULA DE INFORMATICA APLICADA (AIA)
ESTHER MALDONADO, Diplomada en Arquitectura
FERNANDO SUERO, Diplomado en Telecomunicaciones
ALEJANDRO MARCOS, Licenciado en Química

Ediciones Siglo Cultural, S.A.

Dirección, redacción y administración:
Sor Angela de la Cruz, 24-7.º G. Teléf. 279 40 36. 28020 Madrid.

Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28020 Madrid.

Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América. 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

Todos los derechos reservados. Este libro no puede ser, en parte o totalmente, reproducido, memorizado en sistemas de archivo, o transmitido en cualquier forma o medio, electrónico, mecánico, fotocopia o cualquier otro, sin la previa autorización del editor.

ISBN del tomo: 84-7688-020-0
ISBN de la obra: 84-7688-018-9.

Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.

Imprime:
TRAIMSA. Nicolás Morales, 38-40. 28019 Madrid.

© Ediciones Siglo Cultural, S. A., 1986

Depósito legal: M-32.945-1986
Printed in Spain - Impreso en España.

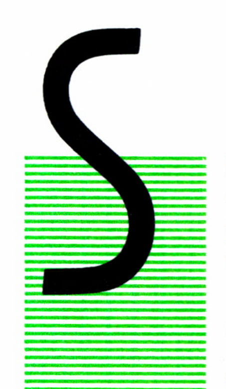
Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Sor Angela de la Cruz, 24-7.° G. Teléf. 279 40 36. 28020 Madrid

Octubre, 1986.
P.V.P. Canarias: 365,-

INDICE

NOCIONES PRELIMINARES 1

DESCRIPCION DE LA PANTALLA



I consideramos la pantalla de nuestro ordenador como un «lienzo» en el que vamos a realizar nuestros dibujos, en primer lugar necesitaremos conocer las dimensiones y características de dicho «lienzo».

Una característica fundamental y común a todos los ordenadores es el hecho de que la pantalla está dividida en «cuadritos» por una retícula. Dicha retícula es inapreciable por el usuario, por lo que conviene conocer sus dimensiones exactas. Estas dimensiones varían de un ordenador a otro, por tanto, vamos a describir las diferentes pantallas que podemos encontrar.

La pantalla de baja resolución en el Spectrum se compone de 22 líneas y 32 columnas, es decir, una retícula de 704 «cuadritos» invisibles.

La memoria del ordenador es capaz de controlar cada una de estas posiciones de la pantalla, para lo cual es necesario identificarlas de alguna manera, de forma que se puedan distinguir entre sí. Esto se resuelve numerando las líneas y las columnas. El Spectrum numera las líneas del 0 al 21 y de arriba a abajo y las columnas de 0 a 31 y de izquierda a derecha, de forma que una posición cualquiera de pantalla vendrá definida por el número de línea y el número de columna en que se encuentra.

Por debajo de la línea número 21 el Spectrum cuenta con dos líneas más, sin embargo, estas dos líneas son para edición de líneas de programa y no pueden usarse para escribir o dibujar.

Además de la retícula de 704 posiciones, el Spectrum puede dividir la pantalla en dos zonas verticales: la zona primera está comprendida entre las columnas 0 y 15 y la zona segunda entre la 16 y 31.

En el caso del Amstrad contamos con muchas más posibilidades, ya que dispone de tres pantallas distintas para baja resolución. Para distinguir estas tres pantallas vamos a numerarlas con 0, 1 y 2.

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

Fig. 1.1. Pantalla de baja resolución del Spectrum.

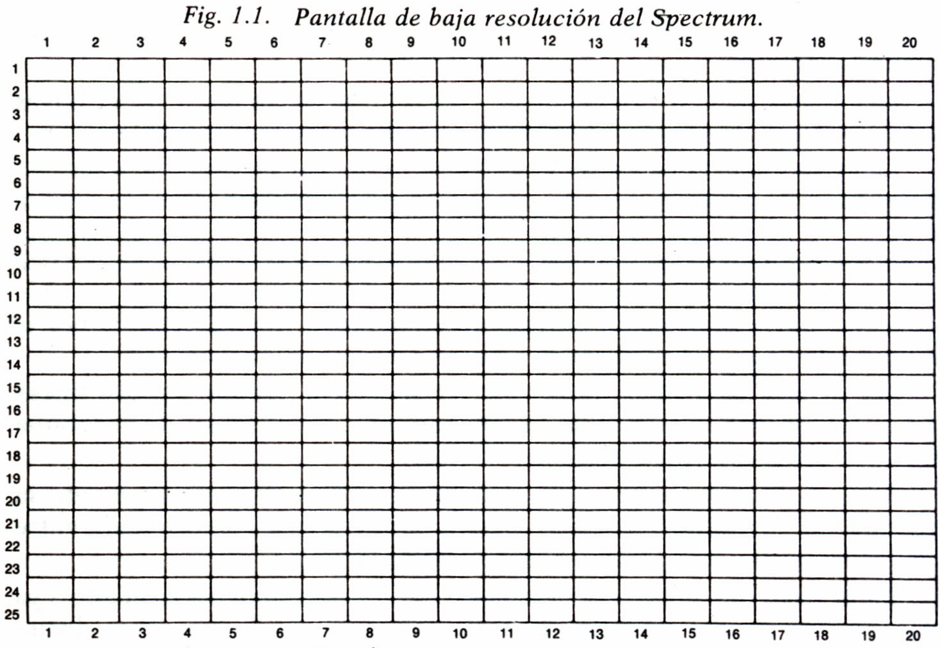


Fig. 1.2. Pantalla de baja resolución del Amstrad en MODE 0.

La pantalla 0 se compone de 25 líneas y 20 columnas, es decir, 500 posiciones distintas. Las líneas están numeradas del 1 al 25 y de arriba a abajo y las columnas del 1 al 20 y de izquierda a derecha.

La pantalla 1 cuenta con 25 líneas y 40 columnas, o lo que es lo mismo, 1000 posiciones distintas. Las líneas están numeradas del 1 al 25 y las columnas del 1 al 40, siempre de arriba a abajo y de izquierda a derecha.

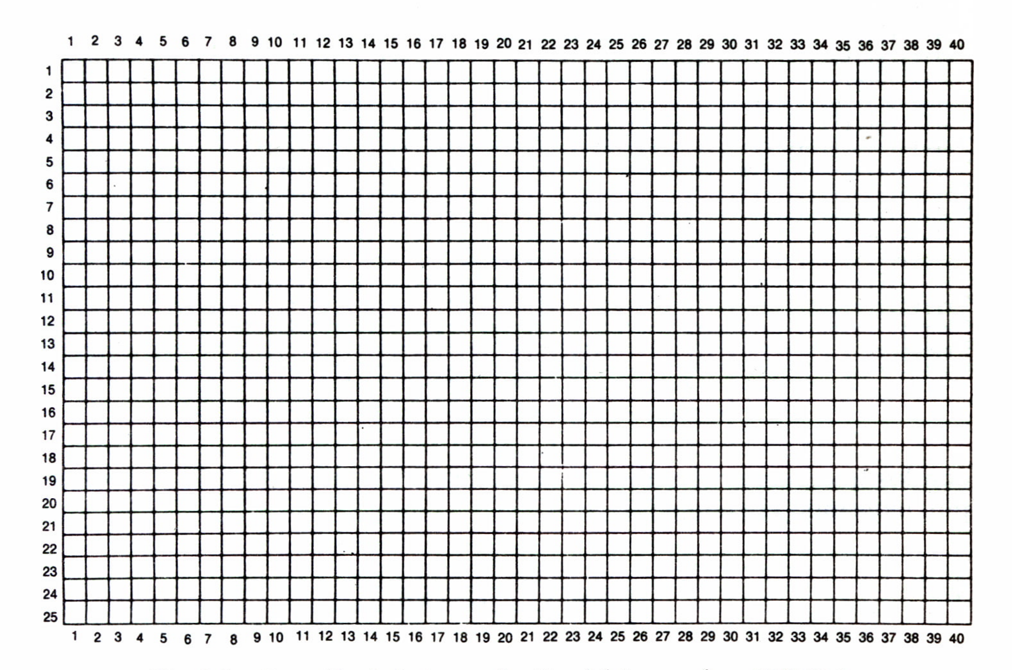


Fig. 1.3. Pantalla de baja resolución del Amstrad en MODE 1.

Por último, la pantalla 2 está dividida en 25 líneas y 80 columnas, es decir, cuenta con 2.000 posiciones distintas. Al igual que en los dos casos anteriores, las líneas van numeradas del 1 al 25 y las columnas del 1 al 80.

El Amstrad, al igual que el Spectrum, también puede dividir la pantalla en zonas verticales, cada una de las cuales abarca como mínimo 13 columnas. Esto significa que la pantalla 0 (20 columnas) sólo dispone de una zona que abarca toda la pantalla. En el caso de la pantalla 1 (40 columnas), ésta queda dividida en tres zonas; las dos primeras abarcan 13 columnas cada una y la última es de 14 columnas. Por último, la pantalla 2 cuenta con seis zonas, todas de 13 columnas, excepto la última, que abarca 15 columnas. Lógicamente, ya que el Amstrad dispone de las tres pantallas descritas, nos interesa saber cómo podemos cambiar de una a otra. Para ello utilizaremos el comando MODE seguido de uno de los números

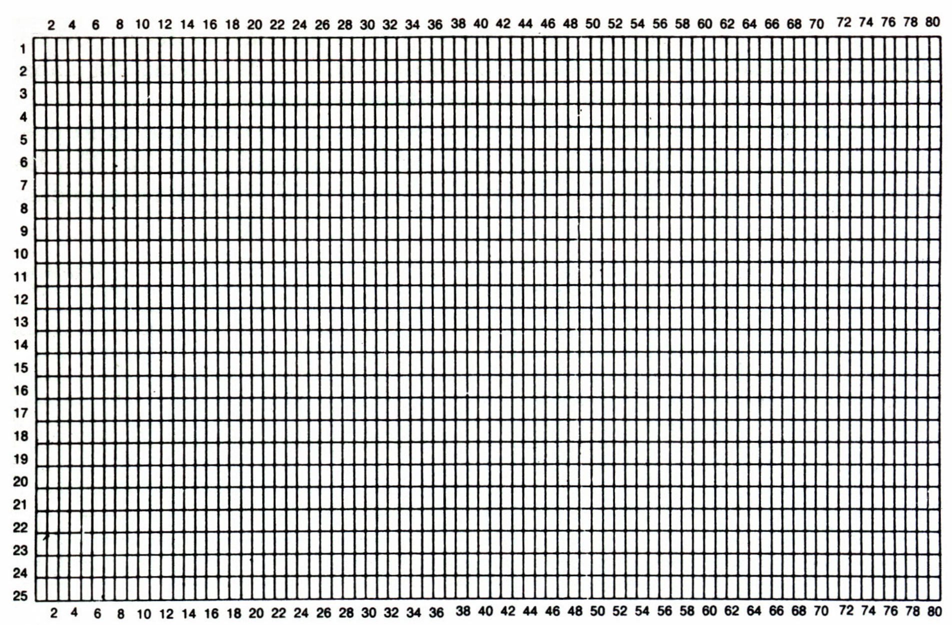


Fig. 1.4. Pantalla de baja resolución del Amstrad en MODE 2.

con los que hemos identificado las distintas pantallas (0, 1, 2). Por ejemplo, tecleando el comando:

MODE 2 ENTER

conseguiremos la pantalla 2 de 80 columnas. También podemos utilizar **MODE** como instrucción, dentro de una línea de programa, de forma que si colocamos diferentes instrucciones **MODE** dentro de un mismo programa, podremos alternar las tres pantallas durante la ejecución:

```
10 REM **********************
20 REM * MODOS DE PANTALLA *
30 REM **********************
40 FOR I=0 TO 2
50 MODE I
60 PRINT "MODOS DE PANTALLA
   EN EL AMSTRAD"
70 PRINT
80 PRINT "ESTE ES EL MODO";I
90 FOR J=1 TO 2000:NEXT
100 NEXT
```

El programa 1.1 tiene por objeto imprimir en pantalla dos frases en los tres modos que existen. El modo va a ir variando en función del valor del índice del bucle **FOR-NEXT**, que se inicia en la línea 40 y se cierra en la 100. La línea 70 imprime una línea en blanco entre las dos frases. En la línea 90 situamos un temporizador, es decir, un bucle que no realiza nada, pero que al repetirse muchas veces consigue retardar la ejecución de la línea del programa siguiente originando una pausa.

En cuanto al Commodore, dispone de una pantalla igual a la del modo 1 del Amstrad, es decir, 25 líneas y 40 columnas, numeradas del 0 al 24 y del 0 al 39, respectivamente, empezando siempre en el ángulo superior izquierdo.

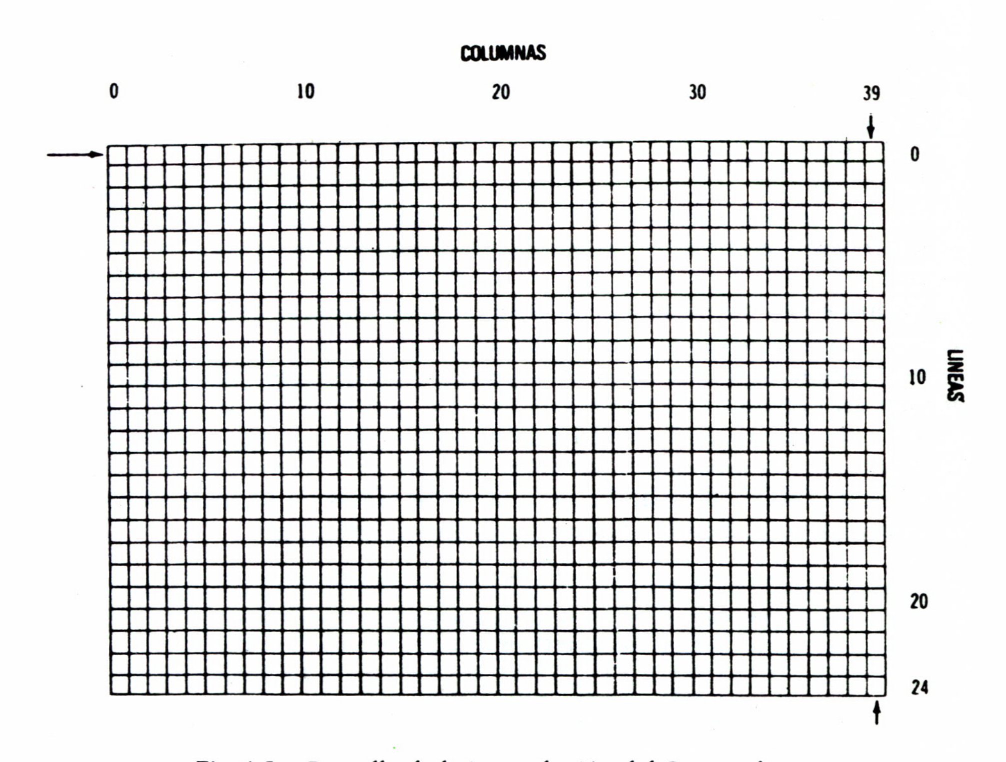


Fig. 1.5. Pantalla de baja resolución del Commodore.

El Commodore dispone de cuatro zonas verticales en la pantalla, cada una de las cuales abarca 10 columnas.

Hasta aquí hemos visto cómo identifica el ordenador la pantalla de baja resolución dentro de su memoria. El siguiente paso es el control de dicha pantalla.

	NUMERO DE LINEAS	NUMERO DE COLUMNAS	NUMERO DE ZONAS	NUMERO DE COLUMNA DE INICIO DE ZONAS
SPECTRUM	22	32	2	0, 16
AMSTRAD	25	20 40 80	1 3 6	1 1, 14, 27 1, 14, 27, 40, 53, 66
COMMODORE	25	40	4	0, 10, 20, 30

Fig. 1.6. Tabla-resumen de las pantallas de baja resolución.



SITUANDONOS EN LA PANTALLA

Ahora que ya sabemos que la pantalla está constituida por una retícula, vamos a ver las diferentes maneras de situarse en una posición determinada de dicha retícula.

La forma mas sencilla de empezar a imprimir en la pantalla es simplemente con la instrucción **PRINT**, tecleando a continuación lo que deseamos imprimir (constantes numéricas, cadenas, operaciones aritméticas o variables).

Si queremos imprimir varias cosas utilizando un solo **PRINT**, tendremos que utilizar los separadores que son el punto y coma (;) y la coma (,).

Separando los datos con punto y coma conseguimos que éstos se impriman en pantalla uno inmediatamente a continuación del precedente, en la misma línea, continuando la impresión en la línea siguiente si se acaba la anterior. Si separamos los datos con comas, el resultado de la impresión es que cada dato aparece en una de las zonas de pantalla explicadas anteriormente, pudiendo abarcar también varias líneas.

```
 10 REM ************************
 20 REM * IMPRESION PANTALLA *
 30 REM ************************
 40 FOR I=0 TO 160
 50 PRINT "*";
 60 NEXT I
 70 PRINT : PRINT
 80 FOR I=1 TO 24
 90 PRINT "*",
100 NEXT I
```

Al ejecutar el programa 1.2 podemos observar todo lo explicado anteriormente. El primer bucle **FOR-NEXT** produce la impresión en pantalla

de varias líneas de asteriscos yuxtapuestos. La línea 70 imprime dos líneas en blanco y el segundo bucle **FOR-NEXT** imprime varias columnas de asteriscos, tantas como zonas tiene la pantalla.

Si lo que queremos es comenzar la impresión en una columna determinada, utilizaremos la función TAB. Esta función va ligada a la instrucción **PRINT** y tiene el siguiente formato:

n de línea PRINT	TAB (columna);	número cadena operación aritmética variable	n

donde *columna* es un número que indica la columna de la pantalla donde va a comenzar la impresión.

Entre barras quedan representadas todas las opciones que podemos imprimir en pantalla. Por último, el exponente *n veces* indica que podemos incluir varias funciones TAB dentro de un mismo PRINT, separándolas unas de otras con punto y coma.

La función TAB es común a los ordenadores de los que se ocupa el presente libro. Sin embargo, podemos matizar un poco más En el Spectrum se puede suprimir los paréntesis de la columna y en el Amstrad podemos suprimir el punto y coma que va a continuación del número de columna.

Si además de controlar la columna en la que comienza la impresión, queremos definir también en qué línea de la pantalla queremos imprimir, podemos utilizar dos funciones: AT en el Spectrum y LOCATE en el Amstrad y el Commodore.

La función AT va ligada al PRINT y tiene el siguiente formato:

n de línea PRINT	AT línea, columna;	número cadena operación aritmética variable	n

donde *línea* es un número entero entre 0 y 21 que indica el número de la línea de pantalla donde se va a realizar la impresión, y *columna* es un número entero entre 0 y 31 que representa la columna de la pantalla a partir de la cual comienza la impresión. Al igual que la función TAB, la función AT puede repetirse varias veces en un PRINT separándolas con punto y coma.

El programa 1.3 divide la pantalla en cuadrados mediante el uso de las funciones TAB y AT.

```
 10 REM ************************
 20 REM *SITUACION EN PANTALLA*
 30 REM ************************
 40 FOR F=0 TO 20
 50 FOR C=0 TO 30 STEP 5
 60 PRINT TAB C;"O";
 70 NEXT C
 80 NEXT F
 90 FOR F=0 TO 20 STEP 5
100 FOR C=0 TO 30
110 PRINT AT F,C;"O";
120 NEXT C
130 NEXT F
```

El resultado es una pantalla como la representada en la figura 1.7.

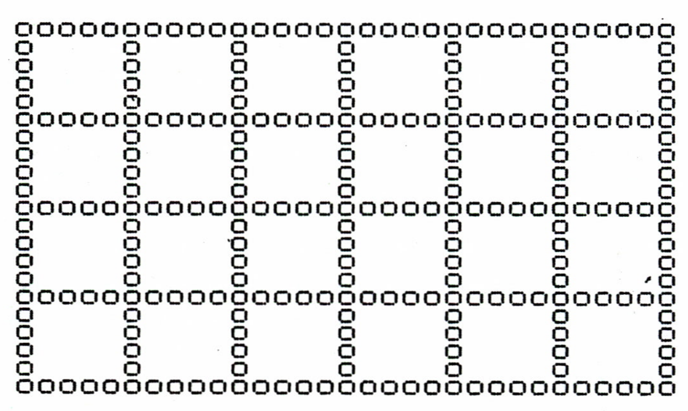


Fig. 1.7. Ejemplo de pantalla utilizando TAB y AT.

La función **LOCATE** tiene la misma misión que **AT**, pero se introduce en el programa de forma distinta, ya que constituye una instrucción independiente del **PRINT** y se sitúa delante de éste.

El formato es el siguiente:

n de línea LOCATE columna, línea

n de línea PRINT | número / cadena / operación aritmética / variable

- número
- cadena
- operación aritmética
- variable

Otra variación importante respecto al **AT** es que primero hay que indicar la columna a partir de la cual se va a realizar la impresión y después

la línea sobre la que deseamos imprimir, es decir, al contrario que en la función AT.

```
10 REM *********************
20 REM * EJEMPLO DE LOCATE *
30 REM *********************
40 CLS
50 LOCATE 17,7
60 PRINT "++++++"
70 LOCATE 19,13
80 PRINT "OO"
90 LOCATE 17,19
100 PRINT "++++++"
```

Tecleando el programa 1.4 y ejecutándolo podemos observar el funcionamiento de LOCATE.

El Amstrad y el Commodore disponen de la función SPC, que tiene por objeto imprimir espacios en blanco en pantalla. Al igual que TAB, debe aparecer en una instrucción PRINT. El formato es de este tipo:

	número
n de línea PRINT SPC (espacios);	cadena
	operación aritmética
	variable

Donde *espacios* es un número que indica el número de espacios que deseamos imprimir. Así, por ejemplo si tecleamos el comando:

```
PRINT SPC (20); "EJEMPLO DE SPC"
```

imprime en pantalla 20 espacios y a continuación la cadena EJEMPLO DE SPC

Por último, conviene saber cómo podemos borrar la pantalla para que siempre podamos «limpiarla» para futuros dibujos. En general, se utiliza la instrucción CLS (que también puede usarse como comando directo); sin embargo, en el Commodore la instrucción correspondiente es: PRINT "♡". Para obtener el carácter situado entre comillas hay que pulsar simultáneamente la tecla CLR/HOME y la tecla SHIFT. Si pulsamos estas dos teclas directamente, es decir, sin incluirlas a continuación de un PRINT, conseguiremos borrar la pantalla en modo directo.

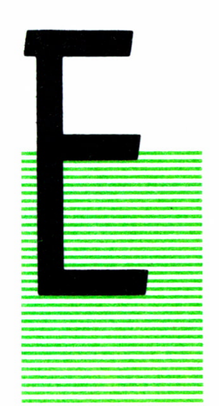
Ya sabemos controlar la impresión en las distintas posiciones de la pantalla; por tanto, ya podemos empezar a realizar nuestros primeros dibujos.

	TAB	AT	LOCATE	SPC
SPECTRUM	*	*		
AMSTRAD	*		*	*
COMMODORE	*		*	*

Fig. 1.8. Tabla-resumen de funciones de situación en pantalla.

EMPEZANDO A DIBUJAR 2

EL JUEGO DE CARACTERES



N un ordenador se denomina carácter a cualquier elemento que pueda visualizarse en una posición de la pantalla. Los ordenadores poseen una serie de caracteres predefinidos que son las letras (mayúsculas o minúsculas), los números y algunos especiales, como el punto, la coma, el asterisco, etc.

Los caracteres pueden ser simplemente pulsados por el usuario para introducirlos en un programa o pueden definirse a través de una serie de funciones especiales. Además, tienen un código numérico, denominado código ASCII que sirve para traducirlos en números.

La función ASC para Amstrad y Commodore, o CODE para Spectrum, sirve para averiguar el código correspondiente a cada carácter. Así, por ejemplo, la letra A mayúscula tiene el número de código 65, mientras que el del asterisco es el 42 para los tres ordenadores.

La función CHR$ es más útil y es común a los tres ordenadores. Es la función inversa de ASC o CODE y sirve para encontrar el carácter asociado a un número de código. Así, por ejemplo, el carácter asociado al código 70 es la letra F mayúscula.

Podemos escribir un programa para cada ordenador que nos permita visualizar los caracteres asociados a los códigos. Serían los siguientes:

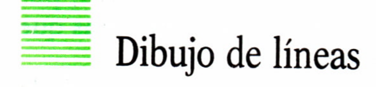
```
10 REM ************************
20 REM * CARACTERES SPECTRUM *
30 REM ************************
40 FOR A=32 TO 255
50 PRINT A;"-";CHR$ (A),
60 NEXT A
```

```
10 REM ******************
20 REM * LOS CARACTERES *
30 REM *  DEL AMSTRAD   *
40 REM ******************
50 MODE 2
60 FOR A=33 TO 191
70 PRINT CHR$(A),
80 NEXT A
```

```
10 REM ******************
20 REM * LOS CARACTERES *
30 REM *  DEL COMODORE  *
40 REM ******************
50 PRINT "♥"
60 FOR A=33 TO 191
70 PRINT CHR$(A),
80 NEXT A
```



DIBUJOS SENCILLOS



Dibujo de líneas

```
* * * * * * * * * * * *
```

Fig. 2.1. Diseño de líneas rectas horizontales.

El dibujo de una recta formada, por ejemplo, por asteriscos es francamente sencillo en baja resolución. Lo único que tenemos que hacer es situar puntos dentro de una misma fila, variando la posición de la columna a lo largo de una recta.

En un ordenador Spectrum, como vimos en el capítulo anterior, existe una función que se denomina AT, que sitúa un carácter en un lugar concreto de la pantalla. Detrás de AT hay que escribir dos números que indican la fila y la columna en las que queremos imprimir.

De esta forma, si queremos dibujar una recta a lo largo de la fila 1, sólo tendremos que visualizarla punto a punto, siempre que todos los puntos tengan el mismo número de fila. Si, por ejemplo, queremos dibujar una línea formada por cinco asteriscos podríamos hacerlo así:

```
10 PRINT AT 1,0;"*"
20 PRINT AT 1,1;"*"
30 PRINT AT 1,2;"*"
40 PRINT AT 1,3;"*"
50 PRINT AT 1,4;"*"
```

Evidentemente, esto es un programa mejorable. En BASIC existe una estructura que hace más fácil el programar tareas repetitivas.

Estas estructuras se denominan bucles y son fundamentales en el diseño de figuras geométricas. Así, por ejemplo, existe un bucle FOR/TO/NEXT en el ordenador Spectrum, en el Amstrad y en el Commodore. Además, en el ordenador Amstrad existe un tipo de bucle diferente denominado WHILE/WEND.

Por consiguiente, el programa anterior podría hacerse de forma más sencilla e incluso ampliarse a 10 asteriscos o más.

```
10 REM ***********************
20 REM * RECTA HORIZONTAL *
30 REM ***********************
40 FOR A=0 TO 10
50 PRINT AT 1,A;"*"
60 NEXT A
```

En el Amstrad podemos situar un asterisco, en un lugar concreto, mediante la palabra LOCATE seguida de dos números que indican la columna y la fila en la que situamos dicho asterisco. Sólo tendríamos que cambiar la línea 50 del programa 2 por la siguiente:

```
50 LOCATE A,1: PRINT "*"
```

No sólo podemos dibujar líneas horizontales, sino también líneas verticales y diagonales gracias a los bucles.

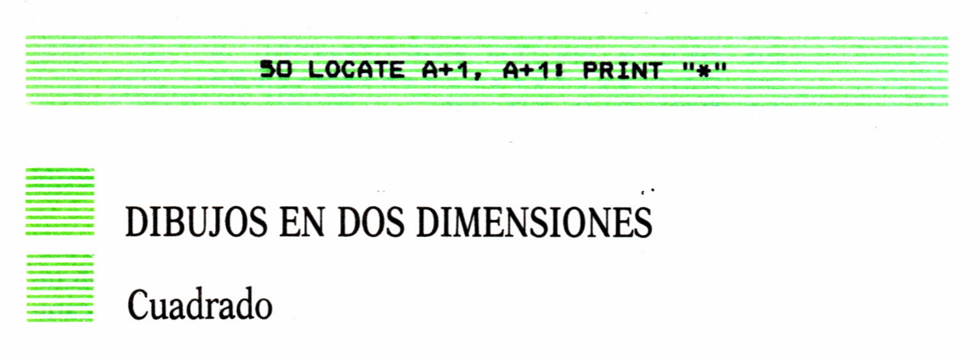
Si queremos dibujar una diagonal formada por 10 asteriscos podríamos hacerlo en el Spectrum del siguiente modo:

```
10 REM ************
20 REM * DIAGONAL *
30 REM ************
40 FOR A=0 TO 9
50 PRINT AT A,A;"*"
60 NEXT A
```



Fig. 2.2. Dibujo de una línea diagonal.

En el Amstrad, lógicamente, tendríamos que escribir la línea 50 utilizando la palabra LOCATE



```
50 LOCATE A+1, A+1: PRINT "*"
```

DIBUJOS EN DOS DIMENSIONES

Cuadrado

Fig. 2.3. Los dibujos en dos dimensiones pueden realizarse utilizando bucles anidados. Este es el caso del cuadrado.

Dibujar en dos dimensiones es tan sencillo como el hacerlo en una.

Si, por ejemplo, queremos dibujar un cuadrado de lado 3, podríamos hacerlo basándonos en el dibujo de 3 líneas rectas, cada una con 3 puntos. Es decir:

```
10 FOR C=1 TO 3
20 PRINT AT 1,C;"."
30 NEXT C
40 FOR A=1 TO 3
50 PRINT AT 2,C;"."
60 NEXT C
70 FOR C=1 TO 3
80 PRINT AT 3,C;"."
90 NEXT C
```

De nuevo observamos que este programa repite varias veces las mismas órdenes. En BASIC podemos agrupar bucles, unos dentro de otros, formando lo que se suele denominar bucles anidados. Estos son muy útiles a la hora de dibujar figuras en dos dimensiones.

Así el programa anterior podría escribirse en el Spectrum de esta manera.

```
10 REM ************
20 REM * CUADRADO *
30 REM ************
40 FOR F=1 TO 3
50 FOR C=1 TO 3
60 PRINT AT F,C;"."
70 NEXT C: NEXT F
```

En el Amstrad y en el Commodore sólo habría que cambiar la línea 60 por la siguiente:

```
60 LOCATE C,F; PRINT "."
```

Podemos también dibujar un cuadrado compuesto por cuadrados negros. Así conseguiremos que la figura no quede hueca. Ello es posible utilizando la función CHR$, que hemos visto anteriormente. Además, pode-

mos hacer el cuadrado más grande sin más que variar el final de los bucles anidados. En el caso del Spectrum el programa sería:

```
10 REM *********************
20 REM * CUADRADO RELLENO *
30 REM *********************
40 FOR F=1 TO 10
50 FOR C=1 TO 10
60 PRINT AT F,C;CHR$ (143)
70 NEXT C: NEXT F
```

En el Amstrad habría que cambiar la línea 60 por la siguiente:

```
60 LOCATE C,F: PRINT CHR$(143)
```

Rectángulo

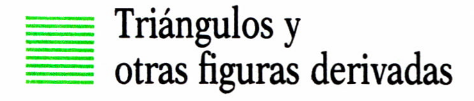
Fig. 2.4. La técnica de dibujo de un rectángulo es igual que la de un cuadrado. Lo único que hay que considerar es que, a diferencia del cuadrado, el rectángulo tiene la base y la altura diferentes.

El dibujo de un rectángulo se basa en la misma técnica de dibujar un cuadrado con la peculiaridad de utilizar dos bucles FOR/TO/NEXT con diferentes valores finales. Así, por ejemplo, si queremos dibujar un rectángulo de 8 por 5 formado por cuadrados negros, para el Spectrum, ejecutaremos el siguiente programa:

```
10 REM ********************
20 REM * RECTANGULO 8x5 *
30 REM ********************
40 FOR A=1 TO 5
50 FOR B=1 TO 8
60 PRINT AT A,B;CHR$ (143)
70 NEXT B: NEXT A
```

Sólo habría que cambiar en el Amstrad y en el Commodore la línea 60 por la siguiente:

```
60 LOCATE B,A; PRINT CHR$(143)
```



Triángulos y otras figuras derivadas

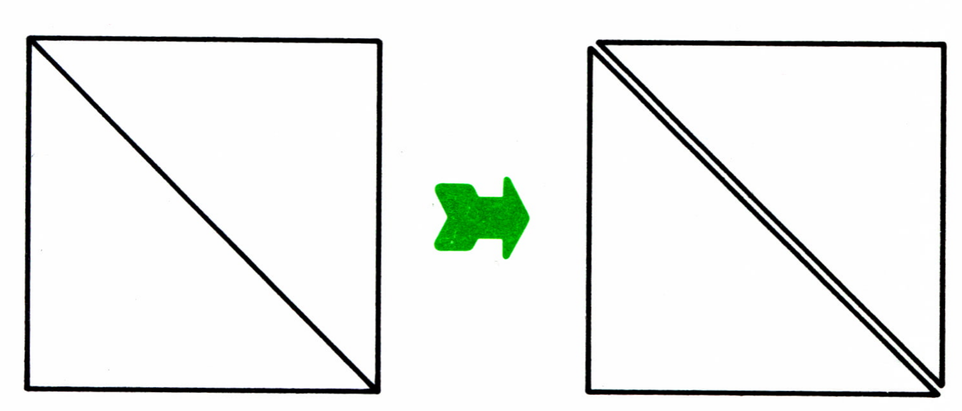


Fig. 2.5. Un triángulo rectángulo puede ser considerado como un cuadrado cortado por la diagonal.

Si pensamos un poco veremos que un triángulo rectángulo no es más que un cuadrado cortado por la diagonal.

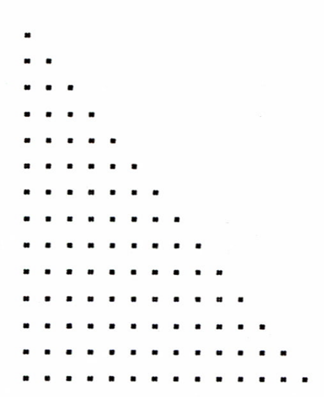


Fig. 2.6. Triángulo rectángulo.

En realidad, la técnica que se sigue para dibujar un triángulo rectángulo es la misma que para dibujar un cuadrado, con la particularidad de que sólo se visualizan aquellos puntos que estén a la derecha o a la izquierda de la diagonal, según queramos un triángulo u otro. Es decir, tendremos que poner una condición que sólo permita dibujar aquellos puntos que estén sólo a la derecha de la diagonal del cuadrado (incluyéndolo). Los puntos de una diagonal son aquéllos cuyo número de fila y de columna coinciden. Los puntos que están a la izquierda de la diagonal tienen un número de fila mayor que el de columna.

De esta forma, si queremos dibujar un triángulo rectángulo apoyado en uno de sus catetos sólo tendríamos que escribir en el Spectrum el siguiente programa:

```
10 REM ************************
20 REM * TRIANGULO RECTANGULO*
30 REM ************************
40 FOR F=1 TO 15
50 FOR C=1 TO 15
60 IF C<F THEN PRINT AT F,C;".
"
70 NEXT C: NEXT F
```

En Amstrad tendríamos que cambiar la línea 60 por la siguiente:

```
60 IF C<F THEN LOCATE C,F:PRINT "."
```

Si quisiéramos dibujar un triángulo invertido sólo deberíamos cambiar la condición en la línea 60.

Para el Spectrum:

```
60 IF C>F THEN PRINT AT F,C;"."
```

Para el Amstrad:

```
60 IF C>F THEN LOCATE C,F:PRINT "."
```

El diseño de una pirámide puede hacerse utilizando una técnica algo más complicada que la empleada para dibujar un triángulo rectángulo. Tenemos que visualizar en la primera fila un punto, en la siguiente tres puntos situados simétricamente respecto del anterior, en la siguiente cinco

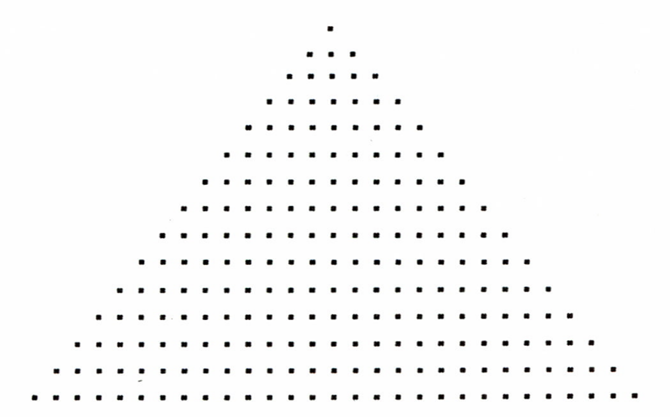


Fig. 2.7. Diseño de una pirámide en dos dimensiones.

puntos y así sucesivamente. Esto se consigue empleando un bucle para las columnas de comienzo y final variables y dependientes de la fila en la que estemos. En el Spectrum sería así:

```
10 REM ************
20 REM * PIRAMIDE *
30 REM ************
40 CLS
50 FOR F=0 TO 14
60 FOR C=14-F TO 14+F
70 PRINT AT F,C;"."
80 NEXT C: NEXT F
```

En el Amstrad sólo habría que cambiar la línea 70 por la siguiente:

```
70 LOCATE C+1, F+1: PRINT "."
```

Basándonos en el diseño de una pirámide, podemos dibujar un rombo resultado de la suma de dos pirámides, con una de ellas invertida.

La técnica de la composición consiste en dibujar el rombo comenzando de forma casi simultánea por sus extremos y llegando hasta el centro como último paso. En realidad, se utiliza una forma «simétrica» de diseño.

En el caso del Spectrum tendríamos que escribir el siguiente programa:

```
10 REM ************************
20 REM * ROMBO PARA SPECTRUM *
30 REM ************************
40 FOR F=0 TO 10
50 FOR C=10-F TO 10+F
60 PRINT AT F,C;"."
70 PRINT AT 20-F,20-C;"."
80 NEXT C: NEXT F
```

Para el Amstrad habría que realizar el programa:

```
10 REM **********************
20 REM * ROMBO PARA AMSTRAD *
30 REM **********************
40 CLS
50 FOR F=0 TO 9
60 FOR C=9-F TO 9+F
70 LOCATE C+1,F+1:PRINT "."
80 LOCATE 19-C,19-F:PRINT "."
90 NEXT C,F
100 LOCATE 1,24
```

FIGURAS DE DIMENSIONES VARIABLES

Una manera de generalizar nuestros diseños es el hacerlos de dimensiones no prefijadas de antemano, es decir, variables. En BASIC existe una palabra para introducir datos variables durante la ejecución de un programa. Esta palabra es INPUT. Si queremos confeccionar un programa que sirva para dibujar un cuadrado de cualquier longitud de lado, tendríamos que hacer en el Spectrum lo siguiente:

```
10 REM ************************
20 REM * CUADRADO LADO VAR. *
30 REM ************************
40 INPUT "LONGITUD DEL LADO "
L
50 FOR F=1 TO L
60 FOR C=1 TO L
70 PRINT AT F,C;"*"
80 NEXT C: NEXT F
```

En el Amstrad habría que cambiar la línea 70 por la siguiente:

```
70 LOCATE C,F:PRINT "*"
```

Además, sería conveniente añadir una línea de borrado de pantalla después de la introducción del dato correspondiente al tamaño del lado, por ejemplo:

```
45 CLS
```

Algo que se recomienda siempre que se escribe un programa es hacerlo a «prueba de bomba». Antes de «presentarlo en sociedad» conviene pensar en las pegas que puede poner nuestro mejor enemigo.

Así, por ejemplo, en el Amstrad, para que nuestro cuadrado parezca realmente un cuadrado, debemos «obligar» al programa a que se ejecute en el tipo de pantalla normal, esto es, la que presenta 40 columnas por 25 filas.

Además, si alguno se pasa de listo y pretende sabotear el programa intentando dibujar un cuadrado que sea mayor que lo que admite la pantalla, el programa no debe caer en la tentación de obedecer, sino que ha de rebelarse y negarse a hacerlo.

Todas estas previsiones se traducen en el siguiente programa para Amstrad:

```
10 REM ******************************
20 REM * CUADRADO A PRUEBA DE BOMBA *
30 REM ******************************
40 MODE 1
50 INPUT "LONGITUD DEL LADO";L
60 IF L<1 OR L>24 THEN PRINT "NO ES POSIBLE":
   GOTO 50
70 CLS
80 FOR F=1 TO L
90 FOR C=1 TO L
100 LOCATE C,F:PRINT "*"
110 NEXT C,F
```

DIBUJO DE CONTORNOS

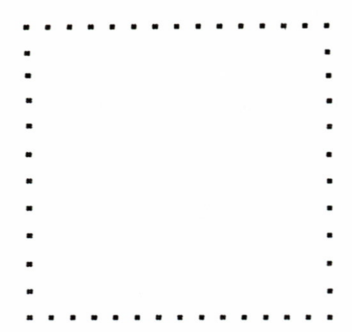


Fig. 2.8. Contorno de un rectángulo. Un contorno supone dibujar sólo los puntos de los extremos.

No sólo podemos dibujar figuras totalmente llenas de puntos o asteriscos. También es posible dibujar el contorno de dichas figuras. Lo único que hay que hacer es situar los puntos sólo en los lados que forman la figura. Ello es posible utilizando un bucle para las columnas que sólo tome los valores extremos. Para ello disponemos de la palabra STEP asociada a los bucles FOR/TO/NEXT, que permite dar el incremento que se desee a la variable de control.

Si, por ejemplo, se desea dibujar el contorno de un rectángulo, hay que considerar tres partes diferenciadas; los dos lados horizontales y los lados verticales. Los dos lados horizontales se dibujan con la misma técnica, por lo que podemos agruparlos en una subrutina. Los lados verticales podemos dibujarlos utilizando la palabra STEP.

En el Spectrum podemos dibujar el contorno de un rectángulo de 15 unidades de ancho por 10 de largo del siguiente modo:

```
 10 REM ************************
 20 REM * CONTORNO RECTANGULAR*
 30 REM ************************
 40 CLS
 50 GO SUB 200
 60 FOR F=1 TO 10
 70 FOR C=1 TO 15 STEP 14
 80 PRINT AT F,C;"."
 90 NEXT C: NEXT F
100 GO SUB 200
110 GO TO 9999
```

```
200 REM * SUBRUTINA DE LADOS *
210 FOR C=1 TO 15
220 PRINT TAB (C);".";
230 NEXT C
240 RETURN
```

Para Amstrad sólo habría que cambiar las líneas 80 y 110 por las siguientes:

```
80 LOCATE C,F: PRINT "."

110 END
```

Otra posibilidad de dibujar contornos es la de utilizar condiciones para visualizar sólo los puntos donde corresponda. Así, si queremos dibujar el contorno de un triángulo rectángulo invertido tendremos que visualizar todos los puntos de la primera fila, todos los de la última columna y aquéllos que forman parte de la hipotenusa, es decir, los que tengan igual número de fila y de columna.

O sea, en el caso del Spectrum, tendríamos que escribir este sencillo programa:

```
10 REM ************************
20 REM * CONTORNO TRIANGULAR *
30 REM ************************
40 CLS
50 FOR F=1 TO 15
60 FOR C=1 TO 15
70 IF F=1 OR C=15 OR F=C THEN
PRINT AT F,C;"."
80 NEXT C: NEXT F
```

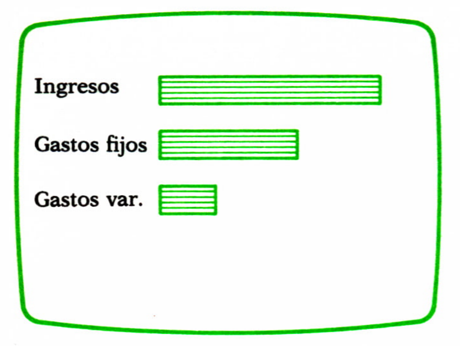
En el Amstrad sólo tendríamos que cambiar la línea 70 por la siguiente:

```
70 IF F=1 OR C=15 OR F=C THEN LOCATE C,F:PRINT"."
```



DIAGRAMAS DE BARRAS

Una de las formas más comunes de representar valores, sobre todo en el mundo comercial, es el de los diagramas de barras, tanto horizontales como verticales.



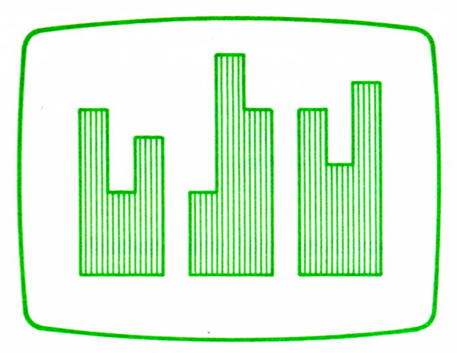


Fig. 2.9. Tipos de diagramas de barras que se dan comúnmente: horizontales y verticales.

Este tipo de representación se utiliza muy frecuentemente en actividades de tipo económico, comercial, estadístico, etc.

Permite, mediante un rápido vistazo, el conocer las variaciones en determinadas actividades. Su importancia es tal que muchos programas de aplicación del tipo de gráficos o de paquetes integrados, o incluso hojas de cálculo, los incluyen.

El diseñar un diagrama de barras, por ejemplo horizontales, es francamente sencillo en BASIC. Consiste en apilar consecutivamente una serie de cuadrados negros, de tal forma que el número de ellos sea igual al de la magnitud a representar, siempre que no se salga de la pantalla.

En primer lugar, desarrollaremos un programa sencillo de representación horizontal de notas. El único «truco» es el utilizar un bucle FOR/NEXT cuyo final sea el valor de la nota. El bucle servirá para repetir el cuadrado negro, cuyo código es el 143 tantas veces como sea necesario.

Para el caso del Spectrum el programa sería:

```
 10 REM ************************
 20 REM * BARRAS HORIZONTALES *
 30 REM ************************
 40 REM * LECTURA DE DATOS *
 50 DIM N(5): DIM N$(5,10)
 60 FOR A=1 TO 5
 70 READ N$(A),N(A)
 80 NEXT A
 90 REM * REPRESENTACION *
100 FOR A=1 TO 5
110 PRINT N$(A),
120 FOR B=1 TO N(A)
130 PRINT CHR$ (143);
140 NEXT B
```

```
150 PRINT : PRINT
160 NEXT A
200 REM * DATOS *
210 DATA "LUIS",8
220 DATA "PEDRO",6
230 DATA "ANA",10
240 DATA "ROSA",5
250 DATA "JUAN",7
```

```
LUIS     ████████
PEDRO    ██████
ANA      ██████████
ROSA     █████
JUAN     ███████
```

Fig. 2.10. Diseño de un gráfico de barras horizontales para el Spectrum. Los datos son los apropiados para no tener que utilizar una transformación a escala.

En el caso del Amstrad vamos a desarrollar un programa algo más complicado. Supongamos que tenemos el caso de una serie de ventas de unos almacenes. Lógicamente, las cifras se saldrían de la pantalla del ordenador.

Habría que representar a escala los resultados. Una forma de hacerlo es buscando el valor máximo de todas las ventas y «llenar» la pantalla con ese valor. Los demás datos se representarían a escala respecto del máximo.

Por tanto, en la primera fase del programa tendríamos que leer los datos. A continuación habría que determinar el máximo de todos ellos para adecuar el resto al tamaño de la pantalla. Por último, deberíamos representar los valores «transformados» como diagramas de barras horizontales.

En definitiva, el programa sería el siguiente:

```
10 REM ************************
20 REM * BARRAS HORIZONTALES *
30 REM *      PARA AMSTRAD     *
40 REM ************************
50 MODE 1
60 REM * LECTURA DE DATOS *
70 FOR TEMPORADA=1 TO 4
80 READ ESTACION$(TEMPORADA),VENTAS(TEMPORADA)
90 NEXT
100 REM * CALCULO DEL MAXIMO *
110 MAXIMO=VENTAS(1)
120 FOR TEMPORADA=2 TO 4
130 IF VENTAS(TEMPORADA)>MAXIMO THEN
```

```
    MAXIMO=VENTAS(TEMPORADA)
140 NEXT
150 REM * TRANSFORMACION A ESCALA *
160 FOR TEMPORADA=1 TO 4
170 BARRA(TEMPORADA)=VENTAS(TEMPORADA)*12/MAXIMO
180 NEXT
190 REM * REPRESENTACION *
200 FOR TEMPORADA=1 TO 4
210 PRINT ESTACION$(TEMPORADA),
220 FOR BARRA=1 TO BARRA(TEMPORADA)
230 PRINT CHR$(143);
240 NEXT
250 PRINT:PRINT:PRINT
260 NEXT
270 REM * DATOS *
280 DATA PRIMAVERA, 50000000
290 DATA    VERANO, 75000000
300 DATA     OTONO, 60000000
310 DATA  INVIERNO,100000000
```

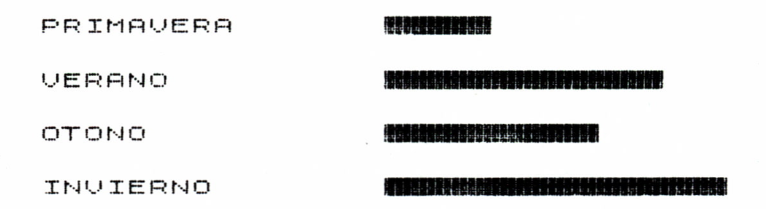


Fig. 2.11. Diseño de un gráfico de barras horizontales para el Amstrad. Debemos transformar los datos dejándolos a escala para poder encajarlos en la pantalla.



OTROS DIBUJOS COMENTADOS

Ahora que ya sabemos cómo diseñar figuras sencillas, estamos en disposición de, mediante algunos pequeños trucos, dibujar figuras algo más complejas, a base de combinar dos o más ya estudiadas.



Bandera de los Estados Unidos

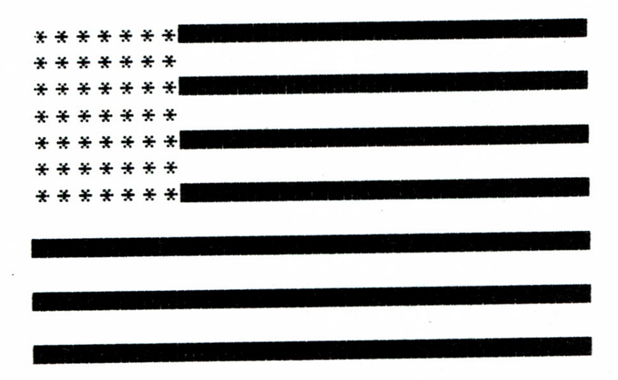


Fig. 2.12. Bandera de los Estados Unidos de América en blanco y negro.

Este ejemplo está compuesto por un cuadrado de estrellas y dos diagramas de barras. Dibujaremos primero un cuadrado de 7 por 7 estrellas; a continuación diseñaremos las barras que están a la derecha de las estrellas; por último, visualizaremos las barras situadas en la parte inferior de la bandera.

El único requisito que hay que cumplir es el situar adecuadamente cada cosa en su sitio.

Para el Spectrum, el programa sería el siguiente:

```
 10 REM ***********************
 20 REM * BANDERA AMERICANA *
 30 REM ***********************
 40 REM * ESTRELLAS *
 50 FOR F=0 TO 6
 60 FOR C=0 TO 6
 70 PRINT AT F,C;"*"
 80 NEXT C: NEXT F
 90 REM * BARRAS PEQUENAS *
100 FOR F=0 TO 6 STEP 2
110 FOR C=7 TO 25
120 PRINT AT F,C;CHR$ (143)
130 NEXT C: NEXT F
140 REM * BARRAS GRANDES *
150 FOR F=8 TO 12 STEP 2
160 FOR C=0 TO 25
170 PRINT AT F,C;CHR$ (143)
180 NEXT C: NEXT F
```

En el caso del Amstrad habría que hacer unas pequeñas modificaciones, con lo que obtendríamos el siguiente programa:

```
10 REM ************************
20 REM * BANDERA USA AMSTRAD *
30 REM ************************
40 MODE 1
50 REM * ESTRELLAS *
60 FOR F=1 TO 7
70 FOR C=1 TO 7
80 LOCATE C,F:PRINT "*"
90 NEXT C,F
100 REM * BARRAS *
110 FOR F=1 TO 7 STEP 2
120 FOR C=8 TO 25
130 LOCATE C,F:PRINT CHR$(143)
140 NEXT C,F
150 REM * BARRAS *
160 FOR F=9 TO 13 STEP 2
170 FOR C=1 TO 25
180 LOCATE C,F:PRINT CHR$(143)
190 NEXT C,F
200 LOCATE 1,24
```

Casita de campo

Como aplicación de lo visto anteriormente (bucles **FOR/TO/NEXT**), vamos a dibujar un esbozo de casa de campo.

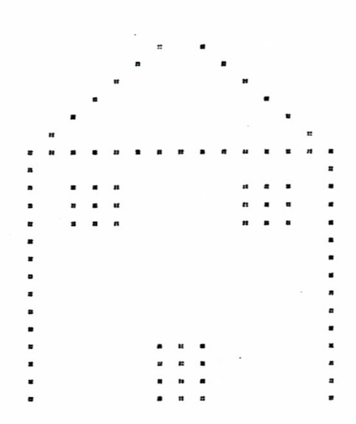


Fig. 2.13. Intento de hacer un dibujo complejo e irregular como es una casa, aunque en modo simplificado.

En general, cuando un dibujo se complica, y sobre todo cuando es irregular, suele utilizarse la técnica de suministrar al programa todos los datos de las coordenadas de todos los puntos mediante la palabra **DATA**. Los programas no son difíciles de comprender, pero tampoco enseñan ninguna técnica nueva. Por ello, vamos a dibujar la casa utilizando las técnicas vistas hasta ahora, es decir, bucles anidados.

En el caso del Spectrum, el programa sería el siguiente:

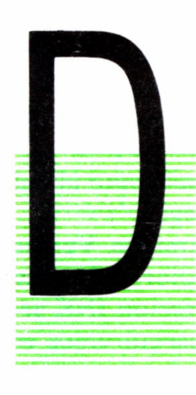
```
 10 REM ********************
 20 REM * MI CASA SPECTRUM *
 30 REM ********************
 40 REM * TEJADO *
 50 FOR F=1 TO 7
 60 FOR C=7-F TO 7+F STEP F*2
 70 PRINT AT F,C;"."
 80 NEXT C: NEXT F
 90 REM * ARMAZON *
100 FOR F=7 TO 21
104 LET D=14
110 LET D=14
120 IF F=7 THEN LET D=1
130 FOR C=0 TO 14 STEP D
140 PRINT AT F,C;"."
150 NEXT C: NEXT F
160 REM * PUERTA *
170 FOR F=18 TO 21
180 FOR C=6 TO 8
190 PRINT AT F,C;"."
200 NEXT C: NEXT F
210 REM * VENTANAS *
220 FOR F=9 TO 11
230 FOR C=2 TO 4
240 PRINT AT F,C;"."
250 NEXT C
260 FOR C=10 TO 12
270 PRINT AT F,C;"."
280 NEXT C: NEXT F
```

Para el Amstrad habría que hacer unas pequeñas modificaciones. El programa sería el siguiente:

```
10 REM ******************************
20 REM * MI CASITA DE CAMPO AMSTRAD *
30 REM ******************************
40 CLS
```

```
50 REM * TEJADO *
60 FOR f=1 TO 8
70 FOR c=9-f TO 9+f STEP 2*f
80 LOCATE c,f:PRINT "."
90 NEXT c,f
100 REM * ARMAZON *
110 FOR f=8 TO 23
120 FOR c=2 TO 16 STEP 14
130 LOCATE c,f:PRINT "."
140 NEXT c,f
150 REM * PUERTA *
160 FOR f=19 TO 23
170 FOR c=7 TO 11 STEP 4
180 LOCATE c,f:PRINT "."
190 NEXT c,f
```

EL MUNDO DE LOS COLORES 3



DADO que la filosofía del coloreado es distinta en cada ordenador, vamos a subdividir el capítulo en dos partes diferenciadas. La primera versará sobre el mundo de los colores para el Spectrum y la segunda tratará sobre el Amstrad. Al final del capítulo daremos unas breves referencias sobre el Commodore. Además, escribiremos algunos programas del capítulo anterior, incluyendo los colores.



LOS COLORES EN EL SPECTRUM

En un ordenador Spectrum disponemos de 8 colores, siempre y cuando, lógicamente, tengamos un monitor en color.

Cada color se corresponde con un código numérico diferente, que es el siguiente:

CODIGO	COLOR
0	NEGRO
1	AZUL
2	ROJO
3	PURPURA
4	VERDE
5	AZUL CLARO
6	AMARILLO
7	BLANCO

Cuando escribimos algo en la pantalla, aunque no lo parezca, existen tres partes que hay que tener en cuenta. Está lo que escribimos, pero además tenemos que considerar dónde escribimos y los bordes de la pantalla.

En resumen, debemos tener en cuenta la «tinta», el «papel» y los «bordes». Para cada una de estas características existe una palabra en BASIC que permite cambiarla de color. Concretamente:

```
INK n       (TINTA)
PAPER n     (PAPEL)
BORDER n    (BORDES)
```

En cada caso el número n indica el número de código de color que deseemos (ver tabla de código).

Cuando se enciende el ordenador, tanto el color del papel como el del borde es blanco (código 7) mientras que el de la tinta es negro.

Podemos, por supuesto, cambiar el color de lo que queramos. Vamos inicialmente, puesto que es lo más intuitivo, a cambiar el color de la tinta con la que escribimos. Ello se consigue mediante la instrucción INK seguida de un número entre el 0 y el 7. Con ello lo que conseguiremos es cambiar el color de lo que escribamos en la pantalla.

Como la mejor forma de aprender a andar es andando, vamos a escribir un programa muy sencillo para visualizar un nombre (por ejemplo, PEPE) en color azul (luego nos iremos complicando la vida poco a poco).

```
10 INK 1
20 PRINT "PEPE"
```

Si ahora queremos listar el programa, veremos que todo el listado aparece en azul. Esto significa que para recuperar el color negro original tendríamos que decírselo al ordenador. Añadamos, pues, una línea al programa.

```
30 INK 0
```

El nombre «PEPE» se ve en azul y el listado en negro. Cada vez que queramos cambiar de color habrá que indicarlo en el programas *antes* de visualizar.

Si quisiéramos ver los 8 colores de la tinta sólo tendríamos que escribir:

```
10 FOR T=0 TO 7
20 INK T
30 PRINT T;" ";"PEPE"
40 NEXT T
50 INK 0
```

Si ejecutamos el programa veremos que aparecen sólo «PEPES» del 0 al 6. El último no aparece sencillamente porque se está escribiendo en color blanco, que es el del papel (todavía no sabemos cómo cambiarlo). Esto es algo parecido a si intentásemos escribir en una pizarra con una tiza negra.

Si queremos ver algo más serio, basta con escribir un programa ligeramente más complicado. Este programa nos permitirá dibujar barras horizontales cada una de un color diferente.

```
 10 REM **********************
 20 REM *     EL MUNDO DE    *
 30 REM *     LOS COLORES    *
 40 REM *   EN EL SPECTRUM   *
 50 REM **********************
 60 FOR A=0 TO 6
 70 INK A
 80 FOR B=0 TO 4
 90 PRINT CHR$ (143);
100 NEXT B
110 PRINT : PRINT
120 NEXT A
130 INK 0
```

Además de cambiar el color de la tinta, podemos cambiar el color del papel. Para ello debemos utilizar la palabra PAPER seguida de un número. Probemos con el siguiente programa:

```
10 FOR A=1 TO 7
20 PAPER A
30 PRINT "PEPE"
40 NEXT A
```

Si lo ejecutamos veremos que realmente no es oro todo lo que reluce. El «papel» sólo cambia de color si se «entera». Si no se escribe algo en él o se borra la pantalla, no se podrá hacer gran cosa. Si escribimos un programa que cada vez que cambie el color del papel borre la pantalla, el efecto será diferente. Como no tenemos una vista de pájaro, incluimos un bucle retardador en la línea 110.

```
  10 REM ********************
  20 REM *COLORES DEL PAPEL*
  30 REM *     SPECTRUM      *
  40 REM ********************
  70 FOR A=1 TO 7
  80 PAPER A
  90 CLS
 100 PRINT AT 10,3;"LOS COLORES
DEL SPECTRUM"
 110 FOR B=1 TO 500: NEXT B
 120 NEXT A
```

También podemos cambiar el color de los bordes de la pantalla. Esto puede conseguirse mediante la palabra BORDER seguida de un número, que será, lógicamente, el del código de color correspondiente.

Si añadimos al programa anterior una línea:

```
5 BORDER A-1
```

veremos el efecto que produce.

Las instrucciones INK y PAPER pueden incluirse también dentro de un PRINT, de este modo el cambio de color de tinta o de papel sólo afectará al conjunto de caracteres que esté dentro de ese PRINT.



EFECTOS ESPECIALES

Con un Spectrum podemos, además, realizar diversos efectos especiales, como son flashes, brillos, sobreimpresiones y vídeos inversos. Para ello existen unas palabras en BASIC que son las siguientes:

```
FLASH n
BRIGHT n
OVER n
INVERSE n
```

Todas ellas llevan un número.

En el caso de FLASH y BRIGHT, este número puede ser 0, 1 u 8. En el caso de INVERSE y OVER sólo puede ser 0 ó 1.



Flash

Si escribimos FLASH 1, conseguiremos que tanto el papel como la tinta se enciendan y se apaguen intermitentemente, produciendo un efecto de parpadeo. Este efecto se anula escribiendo FLASH 0. Tienen vigencia a partir de donde se escriban. Podemos escribir un programa sencillo, por ejemplo:

```
10 CLS
20 FLASH 1
30 PRINT "PEPE"
```

Si ejecutamos el programa veremos que sólo se producen intermitencias en el sitio donde imprimamos el nombre. Para que el ordenador se «entere» de que «todo» debe estar intermitente, habrá que dar una orden después de FLASH 1 que afecte a toda la pantalla. Es decir, lo mejor es hacer:

```
10 FLASH 1
20 CLS
30 PRINT "PEPE"
```

Si queremos desactivar el efecto de parpadeo, utilizaremos la instrucción FLASH 0. Una vez hecho esto hay que «abarcar» toda la pantalla (CLS). Podemos utilizar una espera (PAUSE) para ver el resultado final. Es decir:

```
10 FLASH 1
20 CLS
```

```
30 PRINT "PEPE"
40 PAUSE 100
50 FLASH 30
60 CLS
```

En general todas estas instrucciones de efectos especiales funcionan de igual forma.



Bright

La instrucción BRIGHT 1 hace que el papel aparezca sobreiluminado. Para cancelarlo se emplea BRIGHT 0.



Inverse

INVERSE 1 hace que los puntos que debían estar encendidos se apaguen, y viceversa. Nunca puede afectar a toda la pantalla, sino sólo a los caracteres impresos. Se cancela con INVERSE 0.



Over

OVER 1 permite imprimir dos caracteres en una misma posición de la pantalla, produciéndose el efecto siguiente: si los puntos encendidos del segundo carácter coinciden sobre puntos apagados del primero, dichos puntos se encienden, y si coinciden sobre puntos también encendidos del primer carácter entonces se apagan. Por otra parte, si los puntos apagados del segundo carácter coinciden sobre puntos encendidos del primero, estos puntos permanecen encendidos, y si en ambos caracteres los puntos están apagados, entonces permanecen apagados.

Por último, las instrucciones BRIGHT, FLASH, INVERSE y OVER pueden escribirse como instrucciones independientes en una línea de programa, o pueden incluirse dentro de un PRINT, en cuyo caso sólo afectarán al conjunto de caracteres que vayan a continuación de dicho PRINT.

A continuación escribiremos un programa de demostración de los efectos especiales. Al final del capítulo se incluirán programas vistos en el segundo, pero con color.

```
  10 REM ************************
  20 REM * EFECTOS ESPECIALES *
  30 REM *  PARA EL SPECTRUM   *
  40 REM ************************
  50 LET A$="EFECTOS ESPECIALES
DEL SPECTRUM"
  60 PRINT AT 2,1;A$
  70 PRINT AT 4,13;"NORMAL"
  80 PAUSE 100
  90 BRIGHT 1
  100 PRINT AT 6,1;A$
  110 PRINT AT 8,8;"BRILLO"
  120 PAUSE 100
  130 BRIGHT 0: FLASH 1
  140 PRINT AT 10,1;A$
  150 PRINT AT 12,12;"PARPADEO"
  160 PAUSE 100
  170 FLASH 0: INVERSE 1
  180 PRINT AT 14,1;A$
  190 PRINT AT 16,12;"INVERSO"
  200 PAUSE 100
  210 INVERSE 0: BRIGHT 1: FLASH

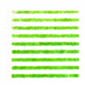
 1
  220 PRINT AT 18,1;A$
  230 PRINT AT 20,7;"BRILLO Y PAR
 PADEO"
  240 BRIGHT 0: FLASH 0
```

ATRIBUTOS DE UN CARACTER

La función ATTR sirve para averiguar la situación de los atributos de una determinada posición de la pantalla. Como resultado devuelve un número decimal entre 0 y 255, que informa de la situación del **FLASH**, BRIGHT, PAPER e INK.

Su sintaxis es:

ATTR (FILA, COLUMNA)



LOS COLORES EN EL AMSTRAD

En un ordenador Amstrad pueden lograrse hasta 27 colores diferentes, lo cual no quiere decir que podemos disponer de ellos al mismo tiempo.

Cada color se corresponde con un código numérico diferente, que es el siguiente:

TABLA MAESTRA DE COLORES

N. TINTA	COLOR TINTA	N. TINTA	COLOR TINTA
0	Negro	14	Azul pastel
1	Azul	15	Naranja
2	Azul brillante	16	Rosa
3	Rojo	17	Magenta pastel
4	Magenta	18	Verde brillante
5	Malva	19	Verde marino
6	Rojo brillante	20	Ciano brillante
7	Púrpura	21	Verde lima
8	Magenta brillante	22	Verde pastel
9	Verde	23	Ciano pastel
10	Ciano	24	Amarillo brillante
11	Azul cielo	25	Amarillo pastel
12	Amarillo	26	Blanco brillante
13	Blanco		

Existen cuatro palabras para poder manipular los diferentes colores. Son las siguientes:

INK
PEN
PAPER
BORDER

Como mínimo hay que asignar dos parámetros a la primera de las palabras (INK). Las otras tres funcionan con un sólo número.

El color del borde (BORDER) funciona «normalmente». El número que se indica coincide, como era de esperar, con el número de código del color cuya tabla hemos incluido anteriormente. Así, por ejemplo, si escribimos:

```
10 BORDER 15
```

al ejecutar el programa, automáticamente cambiará el borde de la pantalla a color naranja, como era previsible.

Pero no todo es tan sencillo como parece. Las instrucciones PEN y PAPER funcionan de un modo algo diferente. El número de código del color no corresponde con el número que se indica detrás de PEN o PAPER. Así, por ejemplo, si escribimos PEN 11 no aparecerá el color azul celeste. Si, además, cambiamos los modos de la pantalla, veremos que cambia el color del siguiente modo:

PEN 11	
MODO 0	COLOR ROSA
MODO 1	COLOR ROJO INTENSO
MODO 2	COLOR AMARILLO INTENSO

Esto se debe a que la cantidad de memoria que se emplea en el almacenamiento de la pantalla no es suficiente para utilizar simultáneamente todos los colores. En definitiva, sólo podemos ver simultáneamente los siguientes colores (dependiendo del modo de escritura).

MODO 0: 16 colores diferentes
MODO 1: 4 colores diferentes
MODO 2: 2 colores diferentes

El número que se escribe detrás de PAPER o PEN no puede ser mayor de 15 (desde 0 hasta 15).

Este número no es un número de color, sino un número de TINTERO. Cuando se enciende el ordenador tenemos la siguiente relación entre colores, plumas y papeles (estos dos últimos serían el número de tintero).

COLOR DE LA TINTA

N. PAPEL/PLUMA	MODO 0	MODO 1	MODO 2
0	1	1	1
1	24	24	24
2	20	20	1
3	6	6	24
4	26	1	1
5	0	24	24
6	2	20	1
7	8	6	24
8	10	1	1
9	12	24	24
10	14	20	1
11	16	6	24
12	18	1	1
13	22	24	24
14	Parpadeo 1,24	20	1
15	Parpadeo 16,11	6	24

Es decir, en modo 1, si escribimos PAPER 11 seguido de CLS, el fondo de la pantalla será de color 6, es decir, rojo intenso.

Podemos observar en la tabla de «plumas» que en modo 2 aparentemente sólo podríamos disponer de los colores azul y amarillo. Esto no es cierto; sólo podemos ver dos colores a la vez, pero éstos pueden ser cualquiera de ellos.

Para ello disponemos de la palabra INK. Detrás de dicha palabra hay que especificar dos números. El primero hace alusión al tintero que se emplea. El segundo al número de código del color. Por ejemplo, si escribimos:

```
10 MODE 1
20 INK 2,26
30 PAPER 2
40 CLS
```

observaremos que el fondo de la pantalla queda de color blanco. Ello se debe a que se ha asignado al tintero 2, utilizado para el papel, el color de código 26, es decir, el blanco brillante.

Normalmente, si no se especifica lo contrario el tintero 0 está asignado al papel y el tintero 1 a la pluma.

Si escribimos:

```
10 INK 0,26

20 INK 1,0
```

Veremos que el papel queda de color blanco brillante y la pluma para escribir de color negro.



LOS COLORES AL AZAR

```
10 REM ***************************
20 REM *VIDRIERA EN EVOLUCION*
30 REM ***************************
40 CLS
50 LET TINTA=INT (RND*7)
60 LET FILA=INT (RND*12)+5
70 LET COLUMNA=INT (RND*23)+4
80 INK TINTA
90 PRINT AT FILA,COLUMNA;CHR$
(143)
100 GO TO 50
```

Esta parte vale tanto para Amstrad como para Spectrum. Existe en ambos una función denominada RND que proporciona como resultado números aleatorios. Puede servirnos en este capítulo para dar colores aleatorios a diversas composiciones. La forma de escribir RND es utilizando la función INT (parte entera) para, de esta forma, obtener como resultado números enteros. Si, por ejemplo, queremos obtener números aleatorios entre el 0 y el 12 y les queremos asignar a la variable a, habremos de escribir:

```
LET A = INT (RND*13)
```

Si nuestro número ha de estar comprendido entre 2 y 14, habrá que escribir:

```
LET A = INT (RND*13) + 2
```

Por ejemplo, si queremos dar colores al azar en el Spectrum (colores entre el 0 y el 7), habrá que escribir:

```
LET A = INT (RND*8)
```

En el Amstrad (entre el 0 y el 26):

```
A = INT (RND*27)
```

En este último hay que tener en cuenta que no todos los colores podrán coexistir.



LOS COLORES EN EL COMMODORE

El Commodore dispone de 16 colores tanto para la tinta como para el papel (fondo) y el borde.

Para variar los colores de la tinta tenemos que pulsar la tecla CTRL al mismo tiempo que un número cualquiera comprendido entre 1 y 8.

De esta manera podemos obtener los ocho primeros colores. Pulsando la tecla «Commodore» a la vez que un número cualquiera entre 1 y 8, obtendremos los ocho colores restantes. Cada uno de estos colores está representado por un carácter especial, tal y como podemos ver en la figura 3.1.

TECLADO	COLOR	PRESENTACION	TECLADO	COLOR	PRESENTACION
CTRL 1	NEGRO		C= 1	NARANJA	
CTRL 2	BLANCO		C= 2	MARRON	
CTRL 3	ROJO		C= 3	ROJO CL.	
CTRL 4	CIAN		C= 4	GRIS 1	
CTRL 5	PURPURA		C= 5	GRIS 2	
CTRL 6	VERDE		C= 6	VERDE CL.	
CTRL 7	AZUL		C= 7	AZUL CL.	
CTRL 8	AMARILLO		C= 8	GRIS 3	

Los 8 primeros colores de la tinta también podemos obtenerlos mediante su código asociado. Si tecleamos el programa 3.11, podemos conseguir una pantalla llena de colores.

La línea 5 sirve para borrar la pantalla y la línea 10 para imprimir cuatro espacios inversos, es decir, cuatro cuadritos del color de la tinta.

```
10 REM ********************
20 REM * CUADROS DE COLOR *
30 REM *     COMMODORE    *
40 REM ********************
50 PRINT CHR$(147)
```

```
60 PRINT CHR$(18);"     ";
70 C=INT(RND(1)*8)+1
80 ON C GOTO 110,120,130,140,150,160,170,180
100 REM * IMPRESION DE COLORES *
110 PRINT CHR$(5);:GOTO 60
120 PRINT CHR$(28);:GOTO 60
130 PRINT CHR$(30);:GOTO 60
140 PRINT CHR$(31);:GOTO 60
150 PRINT CHR$(144);:GOTO 60
160 PRINT CHR$(156);:GOTO 60
170 PRINT CHR$(158);:GOTO 60
180 PRINT CHR$(159);:GOTO 60
```

Por otra parte, la pantalla y el borde también se pueden cambiar de color. Con un **POKE** (véase en el capítulo 4 la explicación detallada de esta instrucción) sobre las direcciones de memoria correspondientes al color de la pantalla y al color del borde, obtendremos los colores deseados entre una gama de 16 (numerados del 0 al 15).

0 NEGRO	8 NARANJA
1 BLANCO	9 MARRON
2 ROJO	10 ROJO CLARO
3 CIAN	11 GRIS 1
4 PURPURA	12 GRIS 2
5 VERDE	13 VERDE CLARO
6 AZUL	14 AZUL CLARO
7 AMARILLO	15 GRIS 3

La dirección de memoria correspondiente al color de borde es la 53280 y la del color del fondo (pantalla) es la 53281. De modo que, por ejemplo, un POKE 53281,8 pondrá la pantalla de color naranja.

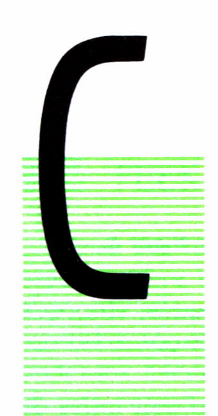
```
  10 REM *********************
  20 REM * RECTANGULO COLOR *
  30 REM *********************
  40 PRINT : PRINT : PRINT : PRI
NT : BORDER 0
  50 GO SUB 200
  60 FOR F=5 TO 15
  70 FOR C=8 TO 23
  80 IF C=8 OR C=23 THEN INK 2:
FLASH 0: PRINT AT F,C;CHR$ (143)
  90 IF C<>8 AND C<>23 THEN INK
1: FLASH 1: PRINT AT F,C;CHR$ (1
43)
 100 NEXT C: NEXT F
```

```
110 GO SUB 200
120 INK 0
130 GO TO 9999
200 REM * SUBRUTINA DE LADOS *
210 FOR C=8 TO 23
220 INK 2: FLASH 0
230 PRINT TAB (C);CHR$ (143);
240 NEXT C
250 RETURN
```

```
 10 REM **********************
 20 REM * BANDERA USA COLOR *
 30 REM *    PARA SPECTRUM    *
 40 REM **********************
 50 REM * ESTRELLAS *
 60 INK 7: PAPER 1
 70 FOR F=0 TO 6
 80 FOR C=0 TO 6
 90 PRINT AT F,C;"*"
100 NEXT C: NEXT F
110 REM * BARRAS *
120 INK 2: PAPER 7
130 FOR F=0 TO 6 STEP 2
140 FOR C=7 TO 31
150 PRINT AT F,C;CHR$ (143)
160 NEXT C: NEXT F
170 FOR F=8 TO 14 STEP 2
180 FOR C=0 TO 31
190 PRINT AT F,C;CHR$ (143)
200 NEXT C: NEXT F
210 INK 0
```

DISEÑANDO FIGURAS NUEVAS





LA MALLA DE 8 POR 8

COMO ya vimos en el capítulo 1, la pantalla del ordenador está formada por una retícula que la divide en «cuadritos» o posiciones. En cada una de estas posiciones podemos imprimir un carácter, es decir, una letra, un número o un signo. Pues bien, para que esto sea posible, cada una de estas posiciones de pantalla está dividida a su vez por una retícula o malla de 8 por 8 puntos, o lo que es lo mismo, cada posición está compuesta de 64 puntos. Cada uno de estos puntos, también llamados pixels, puede tener dos estados: encendido o apagado. Los puntos encendidos son los que realmente se imprimen en pantalla y, por tanto, tendrán el color de la tinta, mientras que los puntos apagados tendrán el color del «papel», es decir, del fondo de la pantalla. Esto quiere decir que cada uno de los caracteres disponibles en el teclado del ordenador tiene que estar definido en una malla de 8 por 8 puntos, para que sea posible su impresión en pantalla.

La figura 4.1 muestra la combinación de puntos encendidos y apagados correspondiente a la definición de la S minúscula en la malla de 8 por 8. Los puntos negros representan los puntos encendidos y los blancos los apagados. Del mismo modo está definido todo el juego de caracteres del teclado de forma que, cuando usamos **PRINT** para imprimir un carácter cualquiera en la pantalla, lo que hace el ordenador es mostrarnos el conjunto de puntos encendidos y apagados que corresponde a dicho carácter, y que él tiene almacenado en su memoria.



EL ORDENADOR POR DENTRO

Para llegar a dominar el tema de los gráficos en ordenador es interesante conocer cómo funciona el ordenador por dentro. Pero no es nece-

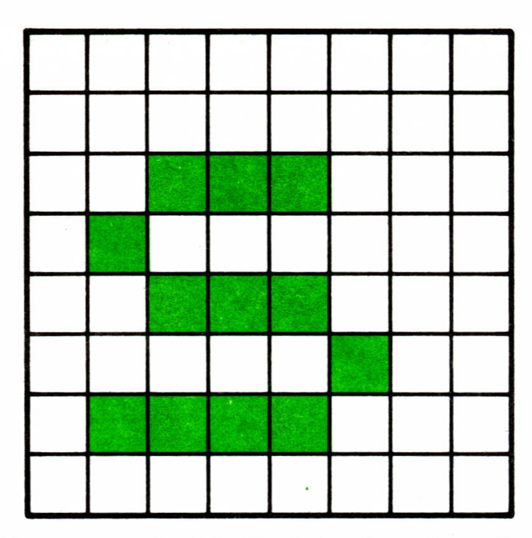


Fig. 4.1. Representación de la S minúscula en la malla de 8 por 8.

sario un estudio exhaustivo del interior de nuestro ordenador, sino que basta con entender un poco cómo se compone la memoria y cómo la controla el ordenador.

La memoria central podríamos asimilarla a un casillero con miles de casillas, en cada una de las cuales podríamos guardar una cosa. En el caso del ordenador cada casilla es lo que se denomina celda de memoria y en cada una de estas celdas podemos almacenar un carácter (letra, número o signo).

La forma en que el ordenador almacena un carácter en una celda de memoria es mediante su código ASCII (del cual hablamos en el capítulo 2). Esto quiere decir que, por ejemplo, la letra A está almacenada en memoria como un 65 que es su código ASCII asociado.

Todos los caracteres predefinidos en el teclado están almacenados de esta forma en la memoria ROM.

Por otra parte, para distinguir una celda de memoria de otra, cada una de ellas tiene asignado un número, que se denomina dirección de memoria. Hay que tener cuidado y no confundir la dirección de memoria con el código ASCII. La primera representa la celda de memoria, mientras que el código representa el dato almacenado en esa celda.

Los programas y datos que introducimos nosotros en el ordenador quedan almacenados en las celdas de la memoria RAM.

Por tanto, hemos visto que la memoria está dividida en dos grandes zonas: la memoria ROM *(Read Only Memory)*, a la cual sólo se puede acceder para «leer» el contenido de la misma; y la memoria RAM *(Random Access Memory)*, que sirve para «leer» y «escribir» información.

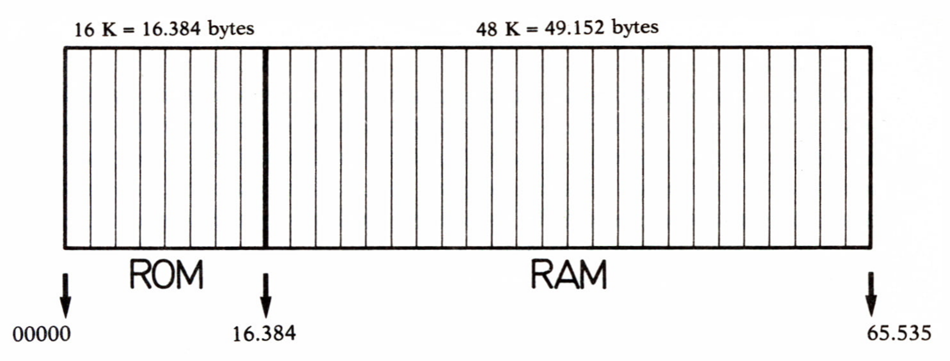


Fig. 4.2. Esquema de memoria del Spectrum de 48 K.



PEEK Y POKE

Estas dos instrucciones están orientadas a darnos información y a manipular sobre las posiciones o celdas de memoria. Pero vamos a estudiar el significado y funcionamiento de cada una de ellas.

PEEK significa en castellano «atisbar», lo que nos da una idea de lo que podemos hacer con esta instrucción. Con PEEK «atisbamos» lo que hay en una determinada posición de memoria definida por su dirección. Esto significa que un PEEK sobre una dirección de memoria cualquiera nos devolverá un número decimal entre 0 y 255. Este número representará una letra, un carácter o un signo (recordemos el código ASCII). Por ejemplo, si tecleamos el comando:

```
PRINT PEEK (18121)
```

En pantalla aparecerá un 0 que representa el dato almacenado en la celda de dirección de memoria 18121.

Para ver lo que hay almacenado en todas las posiciones de memoria RAM del Spectrum sólo tenemos que ejecutar el programa 4.1.

```
10 REM ************************
20 REM * DIRECCIONES MEMORIA *
30 REM ************************
40 PRINT "DIRECCION","CODIGO"
50 PRINT
60 FOR D=16384 TO 65535
70 PRINT D,PEEK D
80 NEXT D
```

Por supuesto, la ejecución entera de este programa ocupará muchísimas pantallas.

En cuanto a POKE,la traducción al castellano sería «hurgar», es decir, que nos dirigimos a una dirección de memoria determinada para insertar en ella un determinado valor. Esto significa que podríamos dirigirnos a una posición de memoria situada dentro de la ROM y alterar su contenido; por tanto, la instrucción POKE es «peligrosa», ya que podríamos modificar sectores de memoria vitales.

Como ejemplo podemos teclear el comando:

```
POKE 18121,65
```

que almacena en la posición de memoria 18121 el número decimal 65. Si ahora tecleamos:

```
PRINT PEEK (18121)
```

En pantalla aparecerá el 65. Si lo que queremos es imprimir en pantalla el carácter correspondiente al código almacenado, tendremos que teclear:

```
PRINT CHR$ (PEEK(18121))
```

y en pantalla aparecerá la letra A.

Hasta aquí hemos hablado un poco de cómo está compuesta la memoria y ya sabemos cómo introducir un dato en una posición de memoria determinada, así que ya podemos pasar a ver el modo de definir nuestros propios caracteres.

DEFINICION DE CARACTERES EN EL SPECTRUM

Aparte de los caracteres definidos en el teclado, nosotros podemos definir caracteres nuevos introduciendo unas pocas instrucciones sencillas en el ordenador.

Estos nuevos caracteres tendrán que estar asignados al teclado para que luego podamos imprimirlos. Las teclas válidas para definir caracteres en el Spectrum son de la A a la U, lo que significa que podemos definir 21

caracteres nuevos. A grandes rasgos, para definir un carácter lo que tenemos que hacer es indicarle al ordenador de alguna manera los puntos encendidos que van a representar dicho carácter, y los puntos apagados en la malla de 8 por 8 puntos ya mencionada.

Dicha malla de 8 por 8 queda definida en la memoria del ordenador por 8 celdas de memoria, también llamadas bytes. Cada uno de estos ocho bytes está compuesto por 8 bits. El bit es la unidad más pequeña de memoria y en él sólo puede almacenarse un 0 o un 1. Esto significa que en un byte habrá almacenada una secuencia de ceros y unos. Esto último parece que está en contradicción con lo que explicamos anteriormente de que en una celda o byte se almacenaba un número decimal comprendido entre 0 y 255. Pero en realidad tiene el mismo significado, ya que una secuencia de 8 cifras compuesta por ceros y unos es la representación de un número decimal en sistema binario.

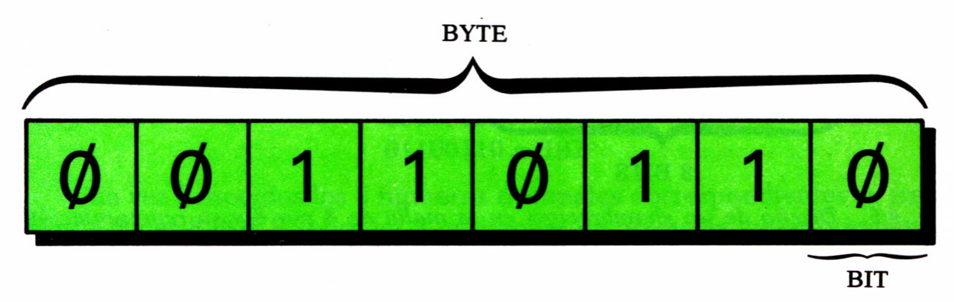


Fig. 4.3. Esquema de una celda de memoria o byte.

En la figura 4.3 está representado el esquema de un byte con sus 8 bits. Para averiguar el número decimal correspondiente al binario almacenado podemos hacer la operación representada en la figura 4.4.

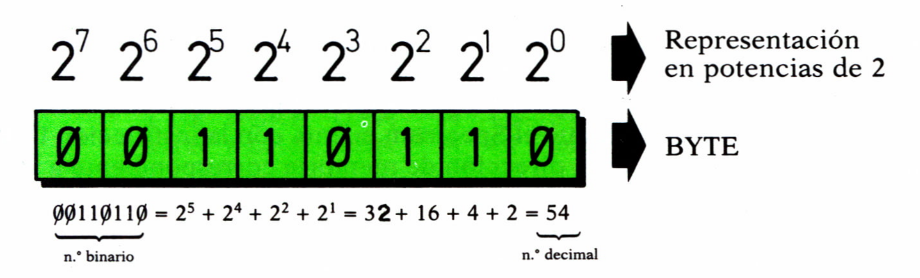


Fig. 4.4. Conversión de sistema binario o decimal.

De todas formas, para mayor comodidad al final del libro se incluye una tabla de conversión decimal-binario.

Visto esto sacamos en conclusión que para definir un carácter nuevo tendremos que introducir 8 números binarios en 8 posiciones de memoria consecutivas. Los números 1 representarán los puntos encendidos que definen el carácter y los números 0 los puntos apagados.

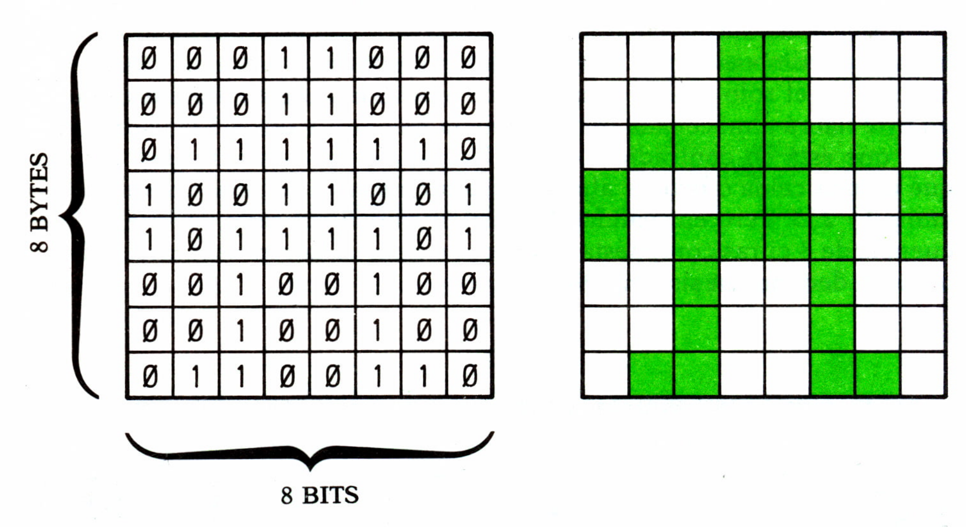


Fig. 4.5. Diseño de un «hombrecito» en la malla de 8 por 8 y su representación binaria.

La figura 4.5 representa los 8 bytes necesarios para definir un «hombrecito», así como el carácter que resultaría.

Hemos dicho anteriormente que cada carácter que definamos lo tenemos que asignar a una tecla comprendida entre la A y la U.

Pero ¿cómo sabemos las 8 posiciones de memoria que corresponden a cada tecla para la definición de caracteres? La zona de memoria RAM reservada para la definición de caracteres es la comprendida entre las direcciones 64349 y 64516, es decir, 168 posiciones de memoria que dividido entre 8 posiciones para cada carácter nos da exactamente los 21 caracteres que podemos definir. Sin embargo no necesitamos recordar estas direcciones de memoria, ya que el Spectrum cuenta con la instrucción USR que se encarga de buscar la dirección de memoria correspondiente a una tecla.

Así, por ejemplo, si queremos saber cuál es la primera dirección de memoria en la que podemos empezar a definir caracteres podemos teclear:

```
PRINT USR "A"
```

y en pantalla aparecerá la dirección 64349 que, junto con las 7 direcciones siguientes forman el conjunto de 8 bytes necesario para definir un carácter en la tecla de la letra A.

Una vez que ya sabemos cómo buscar las direcciones de memoria que nos interesan, tenemos que ver cómo podemos introducir un número binario en cada posición de memoria. Habíamos visto que con la instrucción POKE podíamos introducir un número decimal en una posición de memoria.

Continuando con el ejemplo de la figura 4.5 para introducir el «hombrecito» en la tecla A, tendríamos que transformar los 8 números binarios que lo forman en números decimales. La equivalencia sería la siguiente:

00011000 = 24
00011000 = 24
01111110 = 126
10011001 = 153
10111101 = 189
00100100 = 36
00100100 = 36
01100110 = 102

Una vez conocidos los 8 números decimales correspondientes se trataría de hacer 8 POKE sobre las 8 posiciones de memoria correspondientes a la tecla A. Esto es lo que hace el programa 4.2.

```
 10 REM ************************
 20 REM *   DEFINICION DE UN   *
 30 REM *      HOMBRECITO1     *
 40 REM ************************
 50 POKE USR "A"+0,24
 60 POKE USR "A"+1,24
 70 POKE USR "A"+2,126
 80 POKE USR "A"+3,153
 90 POKE USR "A"+4,189
100 POKE USR "A"+5,36
110 POKE USR "A"+6,36
120 POKE USR "A"+7,102
```

Los POKE que van de la línea 40 a la 110 introducen en las 8 posiciones de memoria correspondientes a la definición de caracteres en la tecla A los 8 números decimales que representan a los 8 binarios que definen el «hombrecito». La función USR se encarga de «buscar» las 8 direcciones de memoria necesarias.

Para poder visualizar a nuestro «hombrecito» en pantalla primero tendremos que ejecutar el programa 4.2. Una vez ejecutado, cada vez que pul-

semos la tecla **A** en modo gráfico (el modo gráfico se obtiene pulsando simultáneamente **CAPS SHIFT** y **GRAPHICS**) aparecerá nuestro hombrecito en la pantalla.

Fig. 4.6. «Hombrecito» definido en la tecla A del Spectrum.

Sin embargo, resulta bastante engorroso tener que transformar números binarios a decimales cada vez que queramos introducir un nuevo carácter en el teclado. Lo ideal sería poder introducir directamente los números binarios que forman el carácter. Esto podemos hacerlo utilizando la función **BIN**. **BIN** se encarga de indicar al ordenador que se va a introducir un número binario en lugar de decimal.

Si tecleamos, por ejemplo, el comando:

```
PRINT BIN00110110
```

Nos imprimirá en pantalla el 54, que es el número decimal equivalente al binario tecleado.

Así que ahora que conocemos la función **BIN**, ya podemos introducir a nuestro «hombrecito» en la tecla **A** del ordenador directamente en código binario.

```
 10 REM ************************
 20 REM *  DEFINICION DE UN    *
 30 REM *      HOMBRECITO2     *
 40 REM ************************
 50 POKE USR "A"+0,BIN 00011000
 60 POKE USR "A"+1,BIN 00011000
 70 POKE USR "A"+3,BIN 01111110
 80 POKE USR "A"+3,BIN 10011001
 90 POKE USR "A"+4,BIN 10111101
100 POKE USR "A"+5,BIN 00100100
110 POKE USR "A"+6,BIN 00100100
120 POKE USR "A"+7,BIN 01100110
```

El programa 4.3 tiene el mismo objetivo que el 4.2, sólo que en este caso los números introducidos en las posiciones de memoria están en código binario.

Si al programa 4.3 le añadimos el programa 4.4, al ejecutarlo conseguiremos una pantalla llena de «hombrecitos» de la mano como la de la figura 4.7.

```
200 REM ************************
210 REM * HOMBRECITOS COGIDOS *
220 REM *       DE LA MANO       *
230 REM ************************
240 FOR F=0 TO 20 STEP 2
250 FOR C=0 TO 31
260 PRINT AT F,C;"🯅"
270 NEXT C
280 NEXT F
```

El primer bucle (línea 230) va contando las líneas de la pantalla de dos en dos y el segundo bucle (línea 240) cuenta las columnas de una en una de modo que mediante la función AT (línea 250) conseguimos que se impriman una línea sí y una no de hombrecitos de la mano.

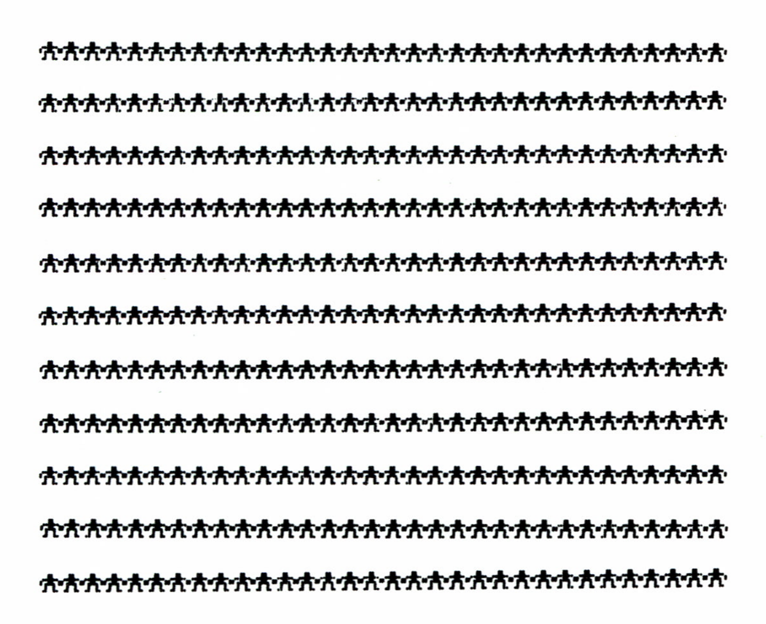


Fig. 4.7. Estos «hombrecitos» de la mano es el resultado de la ejecución del programa 4.4 unido al 4.3.

Una particularidad de los caracteres definidos por el usuario es que no desaparecen de la memoria aunque tecleemos el comando NEW, que borra los programas y datos almacenados en memoria. Por tanto, nuestro «hombrecito» sólo desaparecerá cuando desenchufemos el Spectrum.

También podemos definir una figura más grande utilizando varias teclas, de forma que en cada tecla tengamos un «trozo» de la figura. Impri-

miendo luego en pantalla todos los «trozos» ordenadamente obtendremos una figura más grande.

```
  10 REM *********
  20 REM * BARCO *
  30 REM *********
  40 POKE USR "A"+0,BIN 00000001
  50 POKE USR "A"+1,BIN 00000001
  60 POKE USR "A"+2,BIN 00000001
  70 POKE USR "A"+3,BIN 00001001
  80 POKE USR "A"+4,BIN 00001101
  90 POKE USR "A"+5,BIN 00001001
 100 POKE USR "A"+6,BIN 00001001
 110 POKE USR "A"+7,BIN 00001001
 120 POKE USR "B"+0,BIN 00000000
 130 POKE USR "B"+1,BIN 11100000
 140 POKE USR "B"+2,BIN 00000000
 150 POKE USR "B"+3,BIN 10000000
 160 POKE USR "B"+4,BIN 11000000
 170 POKE USR "B"+5,BIN 11100000
 180 POKE USR "B"+6,BIN 11110000
 190 POKE USR "B"+7,BIN 11111000
 200 POKE USR "C"+0,BIN 00001001
 210 POKE USR "C"+1,BIN 00001001
 220 POKE USR "C"+2,BIN 00001001
 230 POKE USR "C"+3,BIN 11111111
 240 POKE USR "C"+4,BIN 00111111
 250 POKE USR "C"+5,BIN 00011111
 260 POKE USR "C"+6,BIN 00001111
 270 POKE USR "C"+7,BIN 00000111
 280 POKE USR "D"+0,BIN 11111100
 290 POKE USR "D"+1,BIN 11111110
 300 POKE USR "D"+2,BIN 00000000
 310 POKE USR "D"+3,BIN 11111111
 320 POKE USR "D"+4,BIN 11111100
 330 POKE USR "D"+5,BIN 11111000
 340 POKE USR "D"+6,BIN 11110000
 350 POKE USR "D"+7,BIN 11100000
```

El programa 4.5 define en las teclas **A**, **B**, **C** y **D** cuatro caracteres que si los imprimimos juntos representarán la figura 4.8.

Para visualizar el barco de la figura 4.8 en pantalla podemos añadir al programa 4.5 el programa 4.6.

El programa 4.6 pone la pantalla de color azul en la línea 430, y la tinta con la que se van a representar los barcos la pone en blanco en la línea 440.

A continuación sitúa tres barcos en tres posiciones de la pantalla ele-

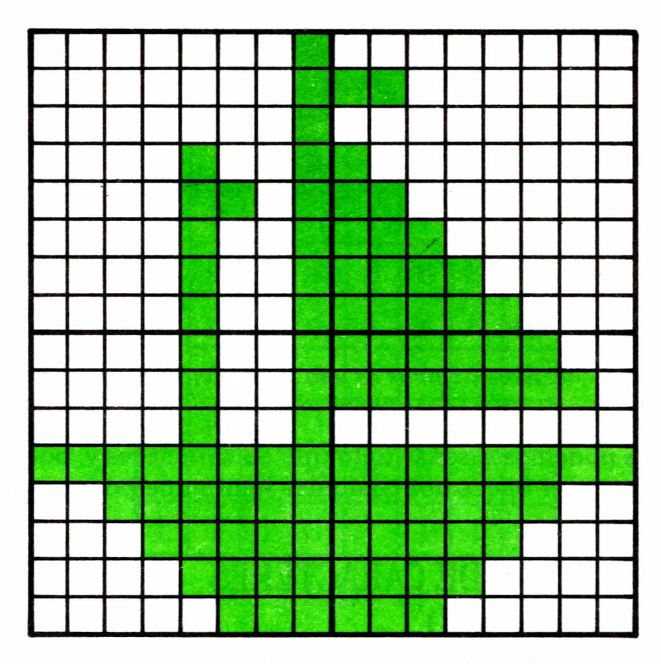


Fig. 4.8. Definición de barco en cuatro caracteres (16 por 16 puntos).

```
400>REM ********************
410 REM * MAR CON BARCOS *
420 REM ********************
430 PAPER 1: CLS
440 INK 7
450 FOR I=1 TO 3
460 LET F=INT (RND*32): IF F>20
THEN GO TO 460
470 LET C=INT (RND*32): IF C>30
THEN GO TO 470
480 PRINT AT F,C;"  "
490 PRINT AT F+1,C;"  "
500 NEXT I
```

gidas al azar en las líneas 460 y 470. El resultado de una posible ejecución será algo similar a la figura 4.9.







Fig. 4.9. Este «mar» con barcos es una posible ejecución del programa 4.6. unido al 4.5.

Así que ya podemos diseñar todo tipo de caracteres y, por tanto, ya podemos dibujar en el Spectrum cualquier figura que podamos imaginar.

DEFINICION DE CARACTERES EN EL AMSTRAD

Para definir caracteres en el **AMSTRAD** existe la palabra **SYMBOL**, en la cual hay que incluir nueve números. El primero es el número del código **ASCII** asociado con el carácter que vamos a definir. Los otros ocho corresponden con los números decimales equivalentes a los binarios de la malla 8 por 8 (ver definición de caracteres en el **SPECTRUM**).

Si, por ejemplo, quisiéramos definir la cara de un fantasma con gafas:

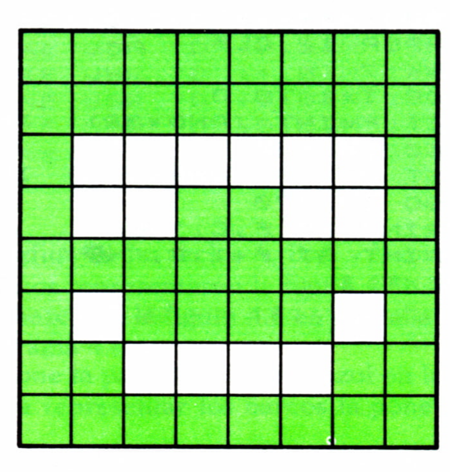


Fig. 4.10. Definición de «fantasma con gafas» en la malla de 8 por 8.

sólo habría que considerar aquellos puntos de la pantalla que estuviesen iluminados. Estos se considerarían «unos» en el sistema binario. Es decir:

128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	Ø	Ø	Ø	Ø	Ø	Ø	1
1	Ø	Ø	1	1	Ø	Ø	1
1	1	1	1	1	1	1	1
1	Ø	1	1	1	1	Ø	1
1	1	Ø	Ø	Ø	Ø	1	1
1	1	1	1	1	1	1	1

Fig. 4.11. Representación binaria del «fantasma con gafas».

Lo único que hay que hacer es multiplicar los «unos» por su correspondiente valor y sumar los resultados para cada fila. Concretamente:

1+2+4+8+16+32+64+128 = 255
1+2+4+8+16+32+64+128 = 255
1+ +128 = 129
1+ +8+16 +128 = 152
1+2+4+8+16+32+64+128 = 255
1+ +4+8+16+32+ +128 = 189
1+2 +64+128 = 195
1+2+4+8+16+32+64+128 = 255

Ahora sólo tenemos que introducir las sumas como números asociados a la instrucción SYMBOL. Es decir:

```
10 SYMBOL 250,255,255,129,153,255,189,195,255
20 PRINT CHR$(250)
```

Cada vez que queramos visualizar por pantalla el diseño que hemos hecho tendremos que escribir la función CHR$ seguida del código numérico

que hayamos asignado a nuestra creación, en la instrucción **SYMBOL** correspondiente (en nuestro caso el número 250).

No sólo podemos definir un carácter, sino que podemos, con grandes dosis de imaginación, construir figuras compuestas por varios caracteres cuyo resultado final, mediante una colocación adecuada, sea un dibujo curioso.

Por ejemplo, podemos diseñar un monstruo (o algo parecido) mediante un programa, definiendo hasta doce caracteres. Lo único que habrá que hacer es situarlos adecuadamente para que compongan la figura deseada.

```
10 REM *************************
20 REM * MONSTRUO DEL INFIERNO *
30 REM *************************
40 MODE 0
50 SYMBOL AFTER 228
60 SYMBOL 228,0,0,16,44,62,28,56,112
70 SYMBOL 229,129,66,126,90,126,106,86,60
80 SYMBOL 230,0,0,8,52,124,56,28,14
90 SYMBOL 231,112,121,127,63,31,7,1,1
100 SYMBOL 232,60,255,255,255,231,215,235,213
110 SYMBOL 233,14,158,254,252,248,224,128,128
120 SYMBOL 234,1,3,15,15,31,63,62,62
130 SYMBOL 235,171,213,235,215,235,255,60,0
140 SYMBOL 236,128,192,240,240,248,252,124,124
150 SYMBOL 237,30,15,15,3,59,127,127,57
160 SYMBOL 238,0,0,0,0,129,195,195,129
170 SYMBOL 239,120,240,240,192,220,254,254,156
180 REM * VISUALIZACION DEL MONSTRUO *
190 LOCATE 8,12:PRINT CHR$(228);CHR$(229);CHR$(230)
200 LOCATE 8,13:PRINT CHR$(231);CHR$(232);CHR$(233)
210 LOCATE 8,14:PRINT CHR$(234);CHR$(235);CHR$(236)
220 LOCATE 8,15:PRINT CHR$(237);CHR$(238);CHR$(239)
```

Podemos redefinir cualquier carácter del ordenador. Podríamos, por ejemplo, diseñar un teclado absolutamente nuevo, de tal forma que cuando escribiéramos una **A**, visualizásemos en pantalla un monstruo o algo parecido.

Lo único que hay que tener en cuenta es que para definir un carácter cuyo código sea inferior al 240 debemos incluirlo en el programa como orden. Ello se consigue mediante la palabra:

SYMBOL AFTER

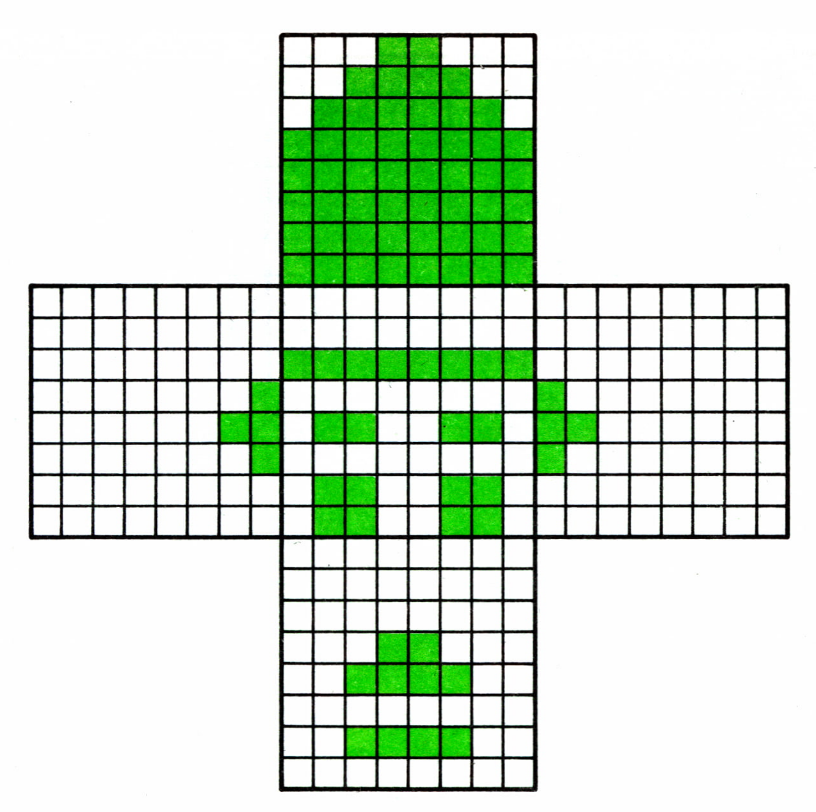


Fig. 4.12. Definición de Charlot utilizando cinco caracteres.

seguida del número de código a partir del cual queramos definir los nuevos caracteres. Por ejemplo, si queremos definir la A para que parezca una B tendremos que escribir:

```
10 SYMBOL AFTER 65
```

seguido de la instrucción de redefinición

```
        20 SYMBOL 65,252,102,102,126,102,102,252,0
   (y   30 PRINT CHR$(65) )
```

Se recomienda, siempre que se desee diseñar un carácter nuevo, el utilizar un papel cuadriculado (y recuadrar 8 filas por 8 columnas).

Probemos ahora a pulsar la A mayúscula a ver qué pasa.

Vamos a experimentar con un programa de este tipo. Antes de que nadie proteste, advertimos que el programa inutilizará nuestro ordenador,

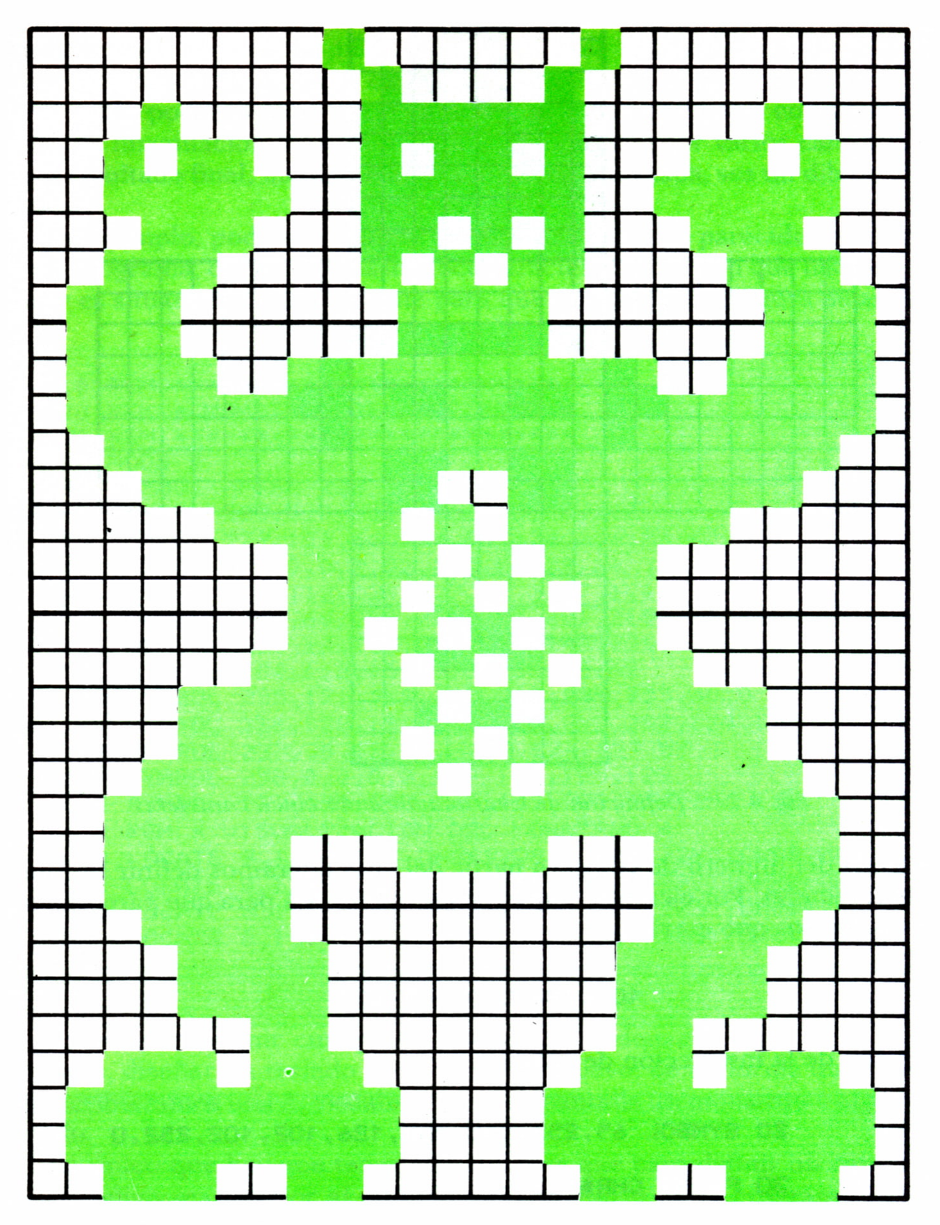


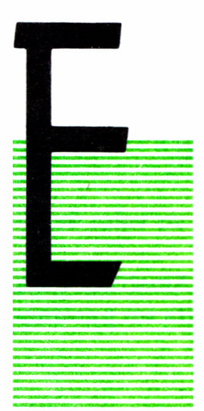
Fig. 4.13. Definición del «monstruo del infierno» utilizando doce caracteres.

salvo que lo apaguemos. Por ello es conveniente que lo grabemos *antes* de ejecutarlo.

No debemos quejarnos. Ya estábamos avisados.

```
10 REM ******************
20 REM * CODIGO SECRETO *
30 REM ******************
40 MODE 0
50 PRINT "JA,JA,JA,...."
60 PRINT:PRINT
70 PRINT "TECLEA FORASTERO..."
80 SYMBOL AFTER 32
90 FOR A=32 TO 127
100 SYMBOL A,INT(RND*25),INT(RND*25),INT(RND*25),
INT(RND*25),INT(RND*25),INT(RND*25),INT(RND*25),INT(
RND*25)
110 NEXT
120 A$=INKEY$:IF A$="" THEN 120
130 INK B,B:B=B+1
140 PEN B
150 IF B>14 THEN B=0
160 PRINT A$;
170 GOTO 120
```

VAMOS A PINTAR, PERO CON BROCHA 5



ESTE capítulo es el último dedicado a la baja resolución, pero esto no significa que sea el final del camino. En los capítulos anteriores hemos aprendido las técnicas fundamentales del manejo de gráficos en baja resolución, y como ejemplo de la potencia de estas técnicas, vamos a construir un programa de diseño gráfico en baja resolución. En suma, vamos a convertir nuestro ordenador en un lienzo donde pintar todo tipo de gráficos multicolores.

El programa consiste en una brocha que se va moviendo por la pantalla dejando un rastro de color de una forma determinada; con ella podremos pintar la forma que queramos, eso sí, dejando un rastro gordo. El programa también dispone de una goma que nos permite borrar una parte del dibujo si no nos gusta, y de una «mano» que nos permitirá movernos por el dibujo sin modificarlo.

Este programa, aunque un poco más complejo que los que hemos visto en capítulos anteriores, nos permitirá acercarnos al mundo del diseño asistido por ordenador. Y, por supuesto, no hay que restringir las opciones del programa a las indicadas, sino que es posible añadir todas las que se nos ocurran para incrementar la potencia del programa.



LA FUNCION INKEY$

Antes de empezar con nuestro programa es necesario que encontremos una forma de poder controlar la brocha desde el teclado, para así moverla a nuestro gusto por la pantalla. En resumen, lo que necesitamos es una forma de poder detectar si hemos pulsado una tecla determinada, y en función de ella realizar una acción u otra.

Para ello disponemos de una función BASIC, INKEY$, que indaga si se

ha pulsado alguna tecla, devolviendo el carácter de la tecla si es cierto que hay alguna pulsada, o devolviendo la cadena vacía, " ", en caso contrario.

Esta función no espera a que pulsemos la tecla, sino que en el momento de ejecutarse mira si hay alguna pulsada. Tampoco devuelve el carácter pulsado en la pantalla.

Por ejemplo, si ponemos la siguiente línea en un programa:

```
10 LET A$ = INKEY$
```

cada vez que el programa pase por ella asignará a A$ el valor de la tecla pulsada en ese momento. Sin embargo, si ponemos esta otra línea:

```
10 LET A$=INKEY$:IF A$="" THEN GOTO 10
```

entonces el programa esperará a que pulsemos una tecla, y su valor se almacenará en A$.

Esta función es válida tanto para Spectrum como para Amstrad. Para Commodore, sin embargo, la función se llama GET, y necesita un parámetro que es una variable alfanumérica.

DISEÑO DE LA BROCHA Y OTRAS HERRAMIENTAS

Como dijimos en la introducción, el programa tiene una brocha, una goma, y una mano. Entonces, ¿por qué no hacer que realmente aparezcan en el programa la brocha, la goma y la mano?

Pues bien, para ello vamos a usar las técnicas aprendidas en el capítulo 4. La brocha, la goma y la mano van a ser 3 caracteres definidos por nosotros.

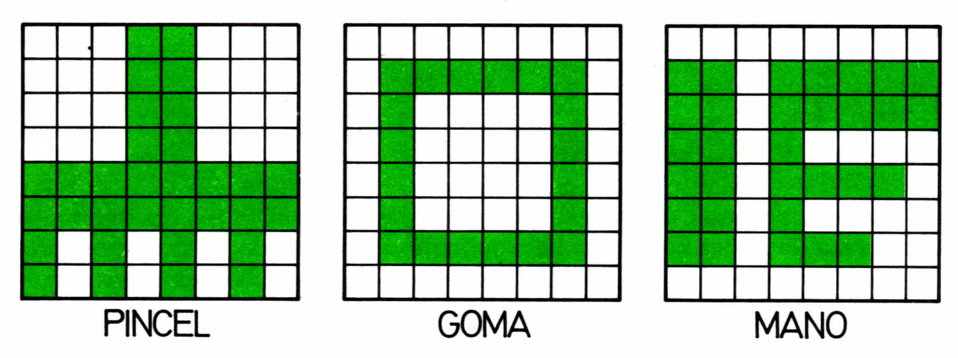


Fig. 5.1. Diseño de brocha, goma y mano en la malla de 8 por 8.

Cada una de estas formas van a ser asignadas a un carácter. Así a la brocha la corresponde la M en el Spectrum, y al carácter cuyo código ASCII es 240 en Amstrad.

Los 8 números de cada carácter son:

BROCHA: 24, 24, 24, 24, 255, 255, 170, 170
GOMA : 0, 126, 66, 66, 66, 66, 126, 0
MANO : 0, 223, 223, 216, 222, 216, 220, 0

Aunque en el programa el trazo dejado por la brocha es un cuadrado, que ya está definido en el juego de caracteres (CHR$(143)), podríamos hacer que el trazo fuera una forma definida por nosotros, como, por ejemplo, un ladrillo.

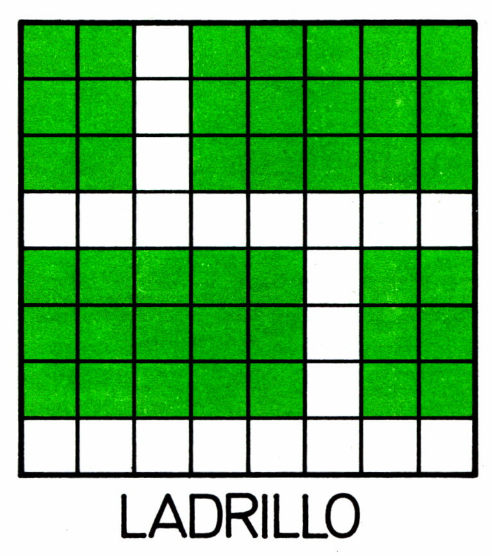


Fig. 5.2. Diseño de ladrillos en la malla de 8 por 8.

Su codificación en este caso es:

LADRILLO: 223, 223, 223, 0, 251, 251, 251, 0

En el programa las herramientas (brocha, etc.) se almacenan en la variable C$, y el rastro dejado por la brocha en la variable B$.

Como recordatorio de la codificación de caracteres gráficos vamos a codificar la brocha.

En el Spectrum:

```
10 FOR I=0 TO 7
15 READ A
```

```
20 POKE USR "B" + I,A
25 NEXT I
30 DATA 24,24,24,24,255,255,170,170
```

Y en el Amstrad:

```
10 SIMBOL 240,24,24,24,24,255,255,170,170
```

A partir de este momento si en el Spectrum escribimos [GRAPHICS] B, o en el Amstrad CHR$ (240), aparecerá en la pantalla el dibujo de la brocha.



MOVIENDO LA BROCHA POR LA PANTALLA

Antes de realizar nuestro programa, vamos a hacer un par de pequeños programas, muy simples, pero que nos permitirán descubrir el principal secreto del programa que haremos más adelante.

La idea es la siguiente: tenemos que mover la brocha por la pantalla mediante el control del teclado. Ya conocemos una función, INKEY$, que nos permite leer el teclado sin molestas esperas o preguntas en la pantalla que nos estropeen el dibujo; ahora lo que tenemos que hacer es escoger cuatro teclas, para así asignarles los cuatro movimientos principales; arriba, abajo, izquierda y derecha. En el caso del Spectrum hemos escogido I, J, K y M, que, como vemos, forman una cruz en el teclado, siendo evidente y simple entender qué dirección significa cada una. Para el Amstrad es aún más sencillo, escogeremos las cuatro teclas de movimiento del cursor.

Ahora lo que necesitamos saber es la situación de la brocha, es decir, su posición en la pantalla; pues bien, la posición vendrá dada por dos variables, X e Y, que nos darán la columna o posición en el eje X, y la fila o posición en el eje Y. Cada vez que desplacemos la brocha mediante una de las teclas, la coordenada correspondiente se actualizará de acuerdo con la dirección tomada.

Sólo hace falta unir todas estas ideas en un programa que lea el teclado, identifique las teclas correspondientes y realice la acción oportuna, de una forma indefinida, por supuesto; así que vamos a emprender manos a la obra.

```
  10 REM ********************
  20 REM * DIBUJERO VER 1. *
  30 REM ********************
  40 CLS
  50 LET X=16: LET Y=11
  60 LET B$=CHR$ (143): PRINT AT
Y,X;B$;
  70 IF INKEY$<>"" THEN GO TO 70
  80 LET A$=INKEY$: IF A$="" THE
N GO TO 80
  90 IF A$="I" THEN LET Y=Y-1
 100 IF A$="M" THEN LET Y=Y+1
 110 IF A$="J" THEN LET X=X-1
 120 IF A$="K" THEN LET X=X+1
 130 PRINT AT Y,X;B$;
 140 GO TO 70
```

```
10 REM **********************
20 REM *   DIBUJERO VER. 1   *
30 REM **********************
40 CLS
50 x=20 : y=12
60 b$=CHR$(143)
70 a$=INKEY$
80 IF a$=CHR$(240) THEN y=y-1
90 IF a$=CHR$(241) THEN y=y+1
100 IF a$=CHR$(242) THEN x=x-1
110 IF a$=CHR$(243) THEN x=x+1
120 LOCATE x,y:PRINT b$;
130 GOTO 70
```

Sencillo, ¿no? En la línea 50 inicializamos los valores de **X** e **Y**; luego asignamos a **B$** el carácter con el que vamos a pintar en la pantalla, y a partir de la línea 70 empieza el bucle de reconocimiento y acción de las teclas. En **A$** está almacenado el valor de la tecla pulsada. Las líneas 70 y 80 del listado correspondiente a Spectrum sirven para evitar un efecto de «metralleta» que se produciría en caso contrario, y si no prueba a quitar la línea 70 y verás lo que pasa. Las líneas 90 a 120 permiten identificar la tecla pulsada. Si ésta coincide con alguna de las de control de la brocha, ésta se desplazará a la posición correspondiente y, finalmente, en la línea 140 volvemos a la línea 70 para comenzar el bucle de nuevo.

Como vemos, el programa es muy sencillo, pero seguramente ya habrá ocurrido el desastre. Si se ha intentado llegar más allá de los bordes de la pantalla, el programa se habrá interrumpido dando un mensaje de error. Esto ocurre porque al intentar pintar con PRINT AT fuera de los límites

de la pantalla el ordenador se «queja», diciendo que está fuera de rango, o sea, fuera de los límites de la pantalla (en el caso del Amstrad y Commodore, LOCATE).

Pero este problema tiene fácil solución; basta con controlar los valores de X e Y, y si éstos coinciden con alguno de los límites de la pantalla, entonces no se actualiza el valor de dicha coordenada.

Esto en el programa se consigue modificando las sentencias 90 a 120, añadiéndoles la condicción de que el valor a actualizar no sea un valor «límite», o sea, añadiendo una condición que sea, aproximadamente, «... y si no estoy en el borde», con lo que ya tenemos el segundo programa:

```
  10 REM *********************
  20 REM * DIBUJERO VER 2. *
  30 REM *********************
  40 CLS
  50 LET X=16: LET Y=11
  60 LET B$=CHR$ (143): PRINT AT
 Y,X;B$;
  70 IF INKEY$<>"" THEN GO TO 70
  80 LET A$=INKEY$: IF A$="" THE
N GO TO 80
  90 IF A$="I" AND Y>0 THEN LET
Y=Y-1
 100 IF A$="M" AND Y<21 THEN LET
 Y=Y+1
 110 IF A$="J" AND X>0 THEN LET
X=X-1
 120 IF A$="K" AND X<31 THEN LET
 X=X+1
 130 PRINT AT Y,X;B$;
 140 GO TO 70
```

```
10 REM **********************
20 REM *   DIBUJERO VER. 2   *
30 REM **********************
40 CLS
50 x=20 : y=12
60 b$=CHR$(143)
70 a$=INKEY$
80 IF a$=CHR$(240) AND y >1 THEN y=y-1
90 IF a$=CHR$(241) AND y<24 THEN y=y+1
100 IF a$=CHR$(242) AND x>1 THEN x=x-1
110 IF a$=CHR$(243) AND x<40 THEN x=x+1
120 LOCATE x,y:PRINT b$;
130 GOTO 70
```

La diferencia de los límites de uno y otro programa es debido a la diferente resolución de las pantallas del Spectrum y Amstrad, que en un caso es de 32 por 22 y en otro de 40 por 24.

Ahora sí parece que todo funciona bien. Este programa es el que nos va a servir como «núcleo» o esqueleto para nuestro programa de dibujo.



EL PROGRAMA

Después de todas las explicaciones anteriores, ya estamos en condiciones de componer nuestro programa y para ello vamos a ir analizándolo desde el principio, poco a poco, para después unir todas las partes en un programa. El listado de todo el programa está al final del capítulo.

La primera parte del programa es la siguiente:

```
10 REM ****************
20 REM * BROCHA GORDA *
30 REM ****************
40 PAPER 7: INK 0: BORDER 7
45 DIM P(32,22)
50 LET X=16: LET Y=11
```

En ella inicializamos los valores de la tinta, el papel y el borde, que, evidentemente, pueden ser cambiados a gusto del usuario. También aquí dimensionamos una matriz llamada P, que abarca las dimensiones de la pantalla y que va a servirnos para guardar y recordar, cuando lo necesitemos, los trazos, en este caso caracteres. Las dimensiones varían de un ordenador a otro por el diferente tamaño de la pantalla en cada caso. Y, por supuesto, inicializamos las coordenadas donde vamos a pintar para que correspondan a las del centro de la pantalla.

En el siguiente bloque inicializamos los caracteres gráficos que nos van a servir para indicar la brocha, la goma y la mano. En el Spectrum los caracteres definidos son la «B» para la brocha, la «G» para la goma y la «M» para la mano. En el Amstrad van a ser los caracteres cuyos códigos corresponden a 240, 241 y 242, respectivamente, para cada una de las herramientas. Los datos para componer los caracteres gráficos están en los DATA del final del programa.

```
   60 FOR I=0 TO 7: READ A: POKE
USR "B"+I,A: NEXT I
   70 FOR I=0 TO 7: READ A: POKE
USR "G"+I,A: NEXT I
   80 FOR I=0 TO 7: READ A: POKE
USR "M"+I,A: NEXT I
```

```
 380 DATA 24,24,24,24,255,255,17
0,170
 390 DATA 0,126,66,66,66,66,126,
0
 400 DATA 0,223,223,216,222,216,
220,0
 380 DATA 24,24,24,24,255,255,17
0,170
 390 DATA 0,126,66,66,66,66,126,
0
 400 DATA 0,223,223,216,222,216,
220,0
```

En la siguiente línea inicializamos la herramienta con la brocha (en C$), y le damos al rastro que deje el valor del cuadrado lleno (en B$). Además, ponemos una variable llamada PMB, que nos va a servir para saber si pintamos (2), borramos (0) o nos movemos (1).

```
  90 LET C$="B": LET B$="■": LET
 PMB=2
```

A continuación lo que hacemos es borrar por completo la pantalla y pintar en el centro la brocha. También inicializamos una variable A$ que va a servirnos para recordar el carácter que había en el lugar donde estuviera la brocha.

```
 100 CLS
 105 PRINT AT Y,X;C$;
```

Seguidamente inicializamos la matriz P para que primeramente contenga espacios. Como es una matriz numérica, lo que hacemos es almacenar el código ASCII del carácter espacio en blanco, que es el 32.

```
 106 FOR I=1 TO 32: FOR J=1 TO 2
2
 107 LET P(I,J)=32: NEXT J: NEXT
 I
```

Después comienza el bucle principal con la lectura del teclado para escoger la tecla y poder actuar más adelante de una forma u otra, como ya vimos en la sección anterior.

```
 130 IF T$="I" AND Y>0 THEN GO S
UB 300: LET Y=Y-1: LET A$=CHR$ P
(X+1,Y+1): LET ATRIB=ATTR (Y,X)
```

```
 140 IF T$="M" AND Y<21 THEN GO
SUB 300: LET Y=Y+1: LET A$=CHR$
P(X+1,Y+1): LET ATRIB=ATTR (Y,X)
 150 IF T$="J" AND X>0 THEN GO S
UB 300: LET X=X-1: LET A$=CHR$ P
(X+1,Y+1): LET ATRIB=ATTR (Y,X)
 160 IF T$="K" AND X<31 THEN GO
SUB 300: LET X=X+1: LET A$=CHR$
P(X+1,Y+1): LET ATRIB=ATTR (Y,X)
```

El siguiente conjunto de sentencias va a controlar las teclas de movimiento de la brocha. Su estructura es la siguiente:

Si se cumple la condición de que la tecla corresponde a una de movimiento del cursor y no estamos en el borde hacia el cual nos vamos a mover, entonces salta a una subrutina. Según estemos pintando, borrando o moviéndonos hacer una cosa u otra, como veremos. Luego actualiza la coordenada correspondiente y guarda el carácter, aunque nos movamos (en A$), así como su color.

La subrutina que empieza en la línea 300 funciona de la siguiente manera:

Si estamos pintando, simplemente situamos B$ en la posición indicada por X e Y y almacenamos su código en la matriz P. Si borramos, entonces pintamos un espacio y si nos movemos pintamos el carácter almacenado en A$, que es el carácter que había antes, con lo que el dibujo queda sin modificar.

```
 110 IF INKEY$<>"" THEN GO TO 11
0

 120 LET T$=INKEY$: IF T$="" THE
N GO TO 120



 300 IF PMB=2 THEN PRINT AT Y,X;
B$;: LET P(X+1,Y+1)=CODE B$
 310 IF PMB=0 THEN PRINT AT Y,X;
" ";: LET P(X+1,Y+1)=32
 320 IF PMB=1 THEN PRINT AT Y,X;
A$;: POKE 22528+Y*32+X,ATRIB
 330 RETURN
```

En las líneas siguientes comprobamos si se han pulsado «T» o «F» que nos van a permitir cambiar el color de la tinta o del fondo, respectivamente. Si ha sido así, el programa salta a la subrutina de la línea 340 (400 en Amstrad), que espera a que pulsemos un número para dar ese valor a la tinta o al fondo.

```
170 IF T$="T" THEN LET TP=1: GO
SUB 340
180 IF T$="F" THEN LET TP=0: GO
SUB 340


 340 IF INKEY$<>"" THEN GO TO 34
0
 350 LET R$=INKEY$: IF R$<"0" OR
 350>LET R$=INKEY$: IF R$<"0" OR
 R$>"7" THEN GO TO 350
 360 IF TP=1 THEN INK VAL R$: RE
TURN
 370 PAPER VAL R$: RETURN
```

Las líneas siguientes controlan las teclas **P**, **B** y **A** que nos van a permitir cambiar la brocha por la goma o la mano en cada caso.

```
 190 IF T$="P" THEN LET PMB=2: L
ET C$="B"
 200 IF T$="B" THEN LET PMB=0: L
ET C$="G"
 210 IF T$="A" THEN LET PMB=1: L
ET C$="M"
```

Más adelante vamos a controlar la tecla **V** que nos permite borrar la pantalla, situando el color del fondo según el valor del fondo activo que esté en ese momento y colocar la brocha otra vez en el centro; y la **S** para salir del programa, después de quitar la brocha de la pantalla, para ver nuestro dibujo completo.

```
 220 IF T$="V" THEN CLS : LET Y=
11: LET X=16: FOR I=1 TO 32: FOR
 J=1 TO 22: LET P(I,J)=32: NEXT
J: NEXT I: LET A$=" ": LET ATRIB
=224
 230 IF T$="S" THEN PRINT AT Y,X
;A$;: POKE 22526+Y*32+X,ATRIB: G
O TO 9999
 280 PRINT AT Y,X;C$;
 290 GO TO 110
```

Finalmente, estas dos líneas lo que hacen es pintar la brocha en la posición correspondiente y volver al principio del bucle.

Y ya está, ya tenemos nuestro programa para dibujar en la pantalla. Hay ligeras diferencias entre los listados de Spectrum y Amstrad, pero la idea principal y el funcionamiento son los mismos.

Vamos a hacer un resumen de las teclas de control del programa:

= Teclas de control de la brocha: I, J, K, M, en el caso del Spectrum, y las teclas de cursor en el Amstrad.

= T y F seguidos de un número cambian el color de la tinta y del fondo, respectivamente. Por ejemplo, T1 pone la tinta al color 1.

= P, B y A controlan la herramienta que estamos utilizando, P pone la brocha, B, la goma, y A, la mano.

= V borra la pantalla y devuelve la brocha al centro.

= S finaliza el programa.

Al comenzar tendrás que esperar un poco antes de empezar a pintar, porque se están inicializando una serie de variables que utiliza el programa.

El programa, tal y como está, es relativamente completo, pero se le pueden añadir mejoras. Vamos a ver algunas sugerencias.

En el Spectrum, por ejemplo, se pueden añadir teclas para poner los bloques que dibujemos en FLASH o aumentar el brillo con BRIGHT, o pintar un bloque encima de otro con OVER, etc.

En el Amstrad una de las ampliaciones al programa más interesantes es la posibilidad de cambiar los colores de los tinteros. Esto se haría con una tecla que luego necesitará dos números, primero, el número de tintero, y luego, el color de la tinta. Para ello te puedes fijar en el funcionamiento de las teclas F y T; también se puede cambiar la resolución de la pantalla.

Para los dos ordenadores se pueden añadir dos teclas G y C, para salvar y coger un dibujo de la cinta o disco. También para los dos es una opción interesante que al principio del programa pusiera en la pantalla todas las teclas de control con una breve explicación de lo que hace cada una.

Como ves, las posibilidades están sólo limitadas por nuestra imaginación.

```
10 REM ******************************
20 REM * PINTURERO DE BROCHA GORDA *
30 REM ******************************
40 INK 0,0 : PAPER 2:BORDER 0:PEN 0:
   COLORF=2:COLORT=0
50 DIM P(40,24,2)
60 X=20:Y=12
70 SYMBOL 240,24,24,24,24,255,170,170
80 SYMBOL 241,0,126,66,66,66,66,126,0
90 SYMBOL 242,0,223,223,216,222,216,220,0
100 C$=CHR$(240):B$=CHR$(143):PMB=2
110 CLS
```

```
120 LOCATE X,Y:PRINT C$;:AT=COLORT:
    AF=COLORF:A$=" "
130 FOR I=1 TO 40
140 FOR J=1 TO 24
150 P(I,J,0)=32:P(I,J,1)=COLORT:
    P(I,J,2)=COLORF
160 NEXT J,I
170 T$=INKEY$:IF T$="" THEN 170
180 MT$=UPPER$(T$)
190 IF MT$=CHR$(240) AND Y>1 THEN GOSUB 400:
Y=Y-1:A$=CHR$(P(X,Y,0)):AT=P(X,Y,1):AF=P(X,Y,2)
200 IF MT$=CHR$(241) AND Y<24 THEN GOSUB 400:
Y=Y+1:A$=CHR$(P(X,Y,0)):AT=P(X,Y,1):AF=P(X,Y,2)

210 IF MT$=CHR$(242) AND X>1 THEN GOSUB 400:
X=X-1:A$=CHR$(P(X,Y,0)):AT=P(X,Y,1):AF=P(X,Y,2)
220 IF MT$=CHR$(243) AND X<40 THEN GOSUB 400:
X=X+1:A$=CHR$(P(X,Y,0)):AT=P(X,Y,1):AF=P(X,Y,2)
230 IF MT$="T" THEN TP=1:GOSUB 500
240 IF MT$="F" THEN TP=0:GOSUB 500
250 IF MT$="P" THEN PMB=2:C$=CHR$(240)
260 IF MT$="B" THEN PMB=0:C$=CHR$(241)
270 IF MT$="A" THEN PMB=1:C$=CHR$(242)
280 IF MT$="V" THEN GOSUB 600
290 IF MT$="S" THEN :PEN AT: PAPER AF:
    LOCATE X,Y:PRINT A$;:LOCATE 1,25:END
300 LOCATE X,Y:PRINT C$;
310 GOTO 170
400 REM * SUBRUTINA *
410 IF PMB=2 THEN LOCATE X,Y:PRINT B$;:P(X,Y,0)=
ASC(B$):P(X,Y,1)=COLORT:P(X,Y,2)=COLORF
420 IF PMB=0 THEN LOCATE X,Y:PRINT " ";:P(X,Y,0)=
    32:P(X,Y,1)=COLORT:P(X,Y,2)=COLORF
430 IF PMB=1 THEN LOCATE X,Y:PEN AT:PAPER AF:
PRINT A$;:PEN COLORT:PAPER COLORF
440 RETURN
500 REM * SUBRUTINA *
510 R$=INKEY$:IF R$<"0" OR R$>"3"
    THEN 510
520 IF TP=1 THEN PEN VAL (R$):
    COLORT=VAL(R$):RETURN
530 PAPER VAL(R$):COLORF=VAL(R$):RETURN
600 REM * SUBRUTINA *
610 CLS
620 Y=11:X=16
630 FOR I=1 TO 40
640 FOR J=1 TO 24
650 P(I,J,0)=32:P(I,J,1)=COLORT:
    P(I,J,2)=COLORF
```

```
660 NEXT J,I
670 A$=" ":AT=COLORT:AF=COLORF
680 Y=12:X=20
690 RETURN
```

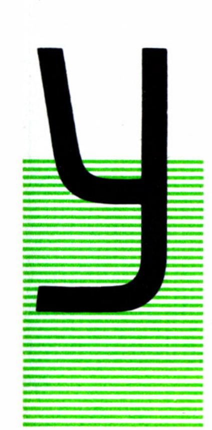
```
  10 REM ****************
  20 REM * BROCHA GORDA *
  30 REM ****************
  40 PAPER 7: INK 0: BORDER 7
  45 DIM P(32,22)
  50 LET X=16: LET Y=11
  60 FOR I=0 TO 7: READ A: POKE
USR "B"+I,A: NEXT I
  70 FOR I=0 TO 7: READ A: POKE
USR "G"+I,A: NEXT I
  80 FOR I=0 TO 7: READ A: POKE
USR "M"+I,A: NEXT I
  90 LET C$="B": LET B$="█": LET
 PMB=2
 100 CLS
 105 PRINT AT Y,X;C$;
 106 FOR I=1 TO 32: FOR J=1 TO 2
2
 107 LET P(I,J)=32: NEXT J: NEXT
 I
 110 IF INKEY$<>"" THEN GO TO 11
0

 120 LET T$=INKEY$: IF T$="" THE
N GO TO 120
 130 IF T$="I" AND Y>0 THEN GO S
UB 300: LET Y=Y-1: LET A$=CHR$ P
(X+1,Y+1): LET ATRIB=ATTR (Y,X)
 140 IF T$="M" AND Y<21 THEN GO
SUB 300: LET Y=Y+1: LET A$=CHR$
P(X+1,Y+1): LET ATRIB=ATTR (Y,X)
 150 IF T$="J" AND X>0 THEN GO S
UB 300: LET X=X-1: LET A$=CHR$ P
(X+1,Y+1): LET ATRIB=ATTR (Y,X)
 160 IF T$="K" AND X<31 THEN GO
SUB 300: LET X=X+1: LET A$=CHR$
P(X+1,Y+1): LET ATRIB=ATTR (Y,X)
 170 IF T$="T" THEN LET TP=1: GO
 SUB 340
 180 IF T$="F" THEN LET TP=0: GO
 SUB 340
 190 IF T$="P" THEN LET PMB=2: L
```

```
ET C$="B"
 200 IF T$="B" THEN LET PMB=0: L
ET C$="G"
 210 IF T$="A" THEN LET PMB=1: L
ET C$="M"
 220 IF T$="V" THEN CLS : LET Y=
11: LET X=16: FOR I=1 TO 32: FOR
 J=1 TO 22: LET P(I,J)=32: NEXT
J: NEXT I: LET A$=" ": LET ATRIB
=224
 230 IF T$="S" THEN PRINT AT Y,X
;A$;: POKE 22528+Y*32+X,ATRIB: G
O TO 9999
 280 PRINT AT Y,X;C$;
 290 GO TO 110
 300 IF PMB=2 THEN PRINT AT Y,X;
B$;: LET P(X+1,Y+1)=CODE B$
 310 IF PMB=0 THEN PRINT AT Y,X;
" ";: LET P(X+1,Y+1)=32
 320 IF PMB=1 THEN PRINT AT Y,X;
A$;: POKE 22528+Y*32+X,ATRIB
 330 RETURN
 340 IF INKEY$<>"" THEN GO TO 34
0
 350 LET R$=INKEY$: IF R$<"0" OR
 R$>"7" THEN GO TO 350
 360 IF TP=1 THEN INK VAL R$: RE
TURN
 370 PAPER VAL R$: RETURN
 380 DATA 24,24,24,24,255,255,17
0,170
 390 DATA 0,126,66,66,66,66,126,
0
 400 DATA 0,223,223,216,222,216,
220,0
```

LA PANTALLA DE ALTA RESOLUCION 6

DESCRIPCION DE LA PANTALLA



A vimos en el capítulo 1 todas las características que presentaba la pantalla de baja resolución. En el presente capítulo vamos a centrarnos en el estudio de la pantalla de alta resolución para que, una vez que la conozcamos, podamos llegar a realizar tantos programas sobre dibujo como ideas se nos ocurran.

La pantalla de alta resolución, al igual que la de baja resolución, está constituida por una retícula que la divide en filas y columnas, pero en este caso dicha retícula está formada por puntos o pixels. Existe una relación entre la pantalla de alta resolución y la de baja ya que cada conjunto de 8 por 8 pixels constituye una posición de pantalla de baja resolución.

Por tanto, para conocer las dimensiones de la pantalla de alta resolución no tenemos más que multiplicar por 8 las dimensiones, ya estudiadas en el capitulo 1, de la baja resolución.

Comenzando por el Spectrum, podemos comprobar que, efectivamente, la pantalla está formada por 176 filas y 256 columnas.

Una característica común a las pantallas de alta resolución del Spectrum y del Amstrad es que están planteadas como unos ejes de coordenadas cartesianas. Esto significa que el origen está en el angulo inferior izquierdo y será el punto 0,0, es decir, columna 0 y fila 0. En el caso del Spectrum, el último punto del eje de abcisas (eje horizontal) será el 255 y el último punto del eje de ordenadas (eje vertical) será el 175.

Por otra parte, el Amstrad dispone de tres modos de pantalla en alta resolución. En el modo 0 se pueden distinguir 160 puntos en horizontal por 200 en vertical. El modo 1 cuenta con 320 puntos en horizontal por 200 en vertical. Por último, el modo 2, que es el de más alta resolución gráfica, distingue 640 puntos en horizontal y 200 en vertical. Sin embargo, para que un mismo programa pueda ser ejecutado en cualquiera de los tres modos, es necesario unificar el sistema de numeración de los puntos. Por este

175

80

100,80

0,0 100 255

Fig. 6.1. Pantalla de alta resolución del Spectrum.

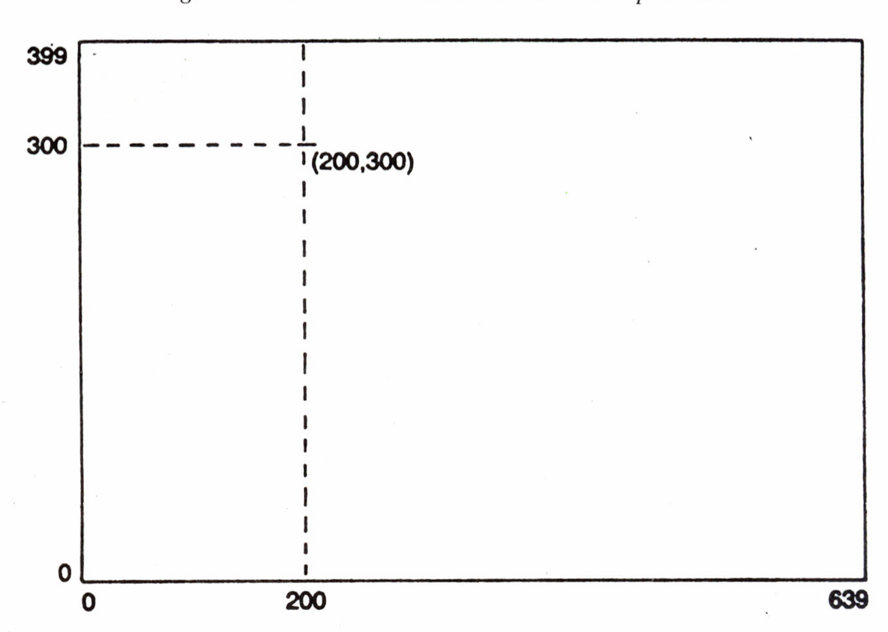


Fig. 6.2. Pantalla de alta resolución del Amstrad.

motivo los ejes de coordenadas están numerados de 0 a 639 en horizontal (abcisas) y del 0 al 399 en vertical (ordenadas).

Esto significa que no es posible representar cada uno de los puntos de este sistema de coordenadas (640*400). En el modo 0 un pixel de la pantalla estará formado por 8 puntos del sistema de coordenadas (4*2). En modo 1 cada pixel constará de 4 puntos (2*2). Por último, en modo 2 un pixel resulta casi imperceptible, ya que abarca sólo 2 puntos (1*2). La figura 6.3 representa de forma esquemática esta idea.

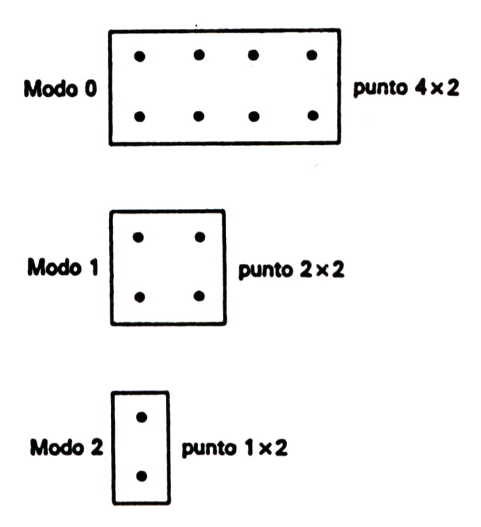


Fig. 6.3. Tamaño de los puntos gráficos en cada modo de pantalla del Amstrad.

La ventaja del modo 0 (menor resolución) sobre el modo 2 (mayor resolución) es que en el modo 0 podemos disponer de 16 colores simultáneamente, mientras que en el modo 1 sólo podemos utilizar 4 colores a la vez y en el modo 2 únicamente 2 colores. Esto ya lo vimos en el capítulo 3 al hablar de los colores en el Amstrad; por tanto, podemos ganar en variedad de colores a costa de perder resolución gráfica, y viceversa, lo que nos permite elegir en cada momento el modo más adecuado.

En cuanto al Commodore, dispone de una pantalla de alta resolución de 320 puntos en horizontal por 200 en vertical. A diferencia de los otros dos ordenadores, el origen de coordenadas está en el punto 0,0 , en el ángulo superior izquierdo. Sin embargo, el Commodore presenta el problema de que no dispone de comandos gráficos BASIC y por tanto, la programación de alta resolución resulta muy compleja. Pero no nos desanimemos, ya que este problema es resoluble puesto que existe en el mercado el BASIC Simon, que es una extensión del BASIC del Commodore, que cuenta con más de 100 comandos BASIC nuevos, entre los cuales están todos los comandos gráficos.

	Número de puntos en horizontal	Número de puntos en vertical
SPECTRUM	256	176
AMSTRAD (640 × 400)	160 320 640	200
COMODORE	320	200

Fig. 6.4. Tabla-resumen de las pantallas de alta resolución.

Una vez conocidas las dimensiones y características de la pantalla de alta resolución, podemos pasar a ver la forma de situarnos en un punto cualquiera de dicha pantalla.



SITUANDONOS EN LA PANTALLA

Vamos a comenzar viendo cómo podemos dibujar un punto en una posición determinada de la pantalla. Para ello disponemos de la instrucción PLOT, que tiene el siguiente formato:

PLOT X,Y, T

donde X indica la coordenada del punto a dibujar en el eje horizontal e Y es la coordenada vertical.

En el caso del Spectrum no podemos añadir el tercer parámetro T, sin embargo, en el Amstrad este parámetro indica el número del color que deseamos para la tinta. Si no indicamos ningún color, el ordenador tomará por defecto el que estuviera definido en el momento de ejecutar la instrucción. En el Commodore el parámetro T sólo puede tomar los valores 0 ó 1, según deseemos que el punto esté encendido (se imprime en pantalla) o apagado.

El Amstrad dispone, además, de otras dos instrucciones bastante útiles para la situación en un punto de la pantalla: MOVE y ORIGIN.

La instrucción MOVE tiene el formato siguiente:

MOVE X,Y

y tiene por objeto mover el cursor gráfico al punto de coordenadas X e Y, pero sin dibujar dicho punto.

La instrucción ORIGIN tiene el mismo formato que MOVE y lo que

hace es trasladar el origen de coordenadas al punto de coordenadas X e Y indicado. De esta forma todos los puntos de la pantalla en lugar de estar referidos al origen del ángulo inferior izquierdo estarán referidos al nuevo origen (X,Y).

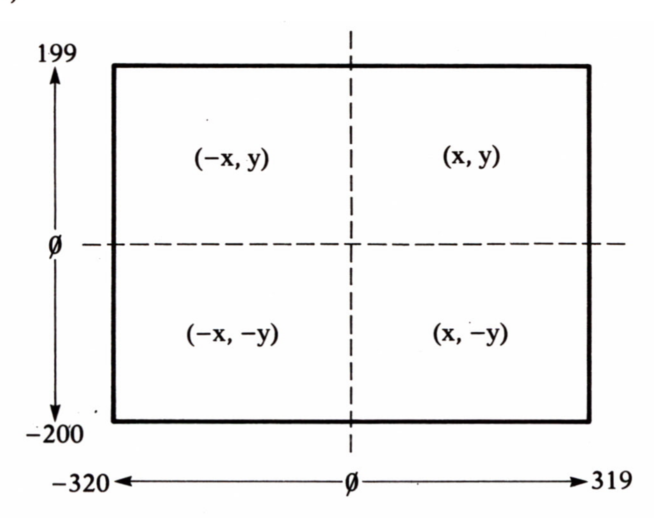


Fig. 6.5. Ejemplo de traslación del origen de coordenadas al centro de la pantalla del Amstrad: ORIGIN 320,200.

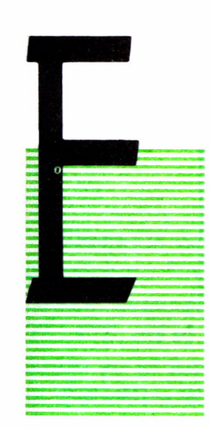
Por otra parte, conviene indicar que el Amstrad admite cualquier parámetro en sus comandos gráficos, aunque estos parámetros queden situados fuera de la pantalla. Evidentemente, si utilizamos parámetros exteriores a las dimensiones de la pantalla (640 por 400), no podremos visualizar nuestros dibujos.

Para borrar la pantalla el Spectrum utiliza la misma instrucción que en baja resolución: CLS. En el Amstrad, en cambio, CLS sólo borra las pantallas de texto y manda el cursor al origen, mientras que para borrar las pantallas gráficas y mandar el cursor gráfico al origen existe la instrucción CLG.

En el Commodore la instrucción HIRES T,F permite la visualización de la pantalla gráfica poniendo la tinta con el color especificado en T y el fondo con el color F. Para anular esta instrucción y pasar de nuevo a la pantalla de texto existe la instrucción NRM.

Ahora que ya sabemos situarnos en la pantalla de alta resolución podemos pasar ya a la realización de los primeros dibujos.

PUNTOS, RECTAS Y CIRCUNFERENCIAS 7



EN el capítulo 6 ya hemos visto cómo podemos situar un punto en la pantalla de alta resolución. Ahora vamos a pasar a dibujar rectas, circunferencias y arcos de circunferencia para, más adelante, aprender a representar las primeras figuras sencillas.

Para representar un punto en la pantalla utilizamos la instrucción PLOT, indicando a continuación las coordenadas X e Y donde queremos imprimir el punto. Recordemos que en alta resolución el origen de coordenadas suele estar en el ángulo inferior izquierdo, no como en baja resolución, que está en el ángulo superior izquierdo.

```
10 REM *********************
20 REM * PANTALLA DE PUNTOS *
30 REM *********************
40 FOR X=0 TO 255 STEP 5
50 FOR Y=0 TO 175 STEP 5
60 PLOT X,Y
70 NEXT Y
80 NEXT X
```

El programa 7.1 muestra un ejemplo del funcionamiento de la instrucción PLOT, llenando la pantalla de puntos.

Sin embargo, como nuestro objetivo es la realización de dibujos con el ordenador, resulta interesante aprender a trazar rectas.

Para trazar rectas el Spectrum dispone de la instrucción DRAW y el Amstrad de la instrucción DRAWR. Ambas instrucciones funcionan igual y su formato es el siguiente:

N. de línea DRAW / DRAWR (incremento x, incremento y)

donde *incremento x* es el aumento de la recta en proyección horizontal e *incremento y* es el aumento en proyección vertical. Esta idea queda representada en la figura 7.1.

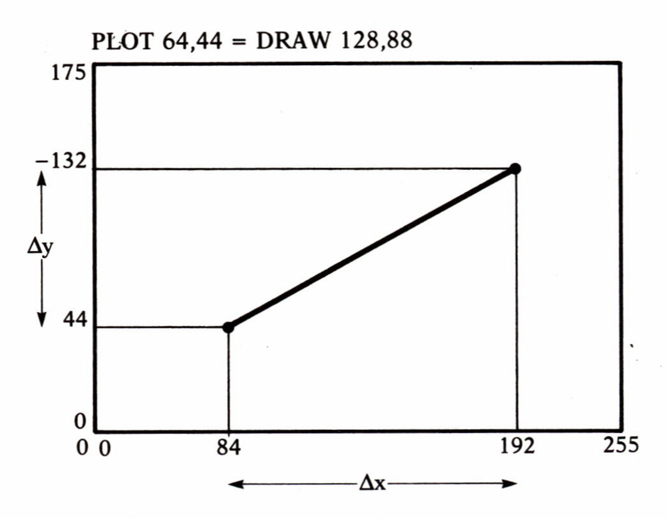


Fig. 7.1. Esquema de funcionamiento de las instrucciones DRAW o DRAWR.

Ahora vamos a trazar nuestra primera línea, que va desde el ángulo inferior izquierdo hasta el ángulo superior derecho, es decir, una diagonal de la pantalla. Para ello tenemos que tener en cuenta las dimensiones de la pantalla del ordenador. En el Spectrum teclearemos el comando:

```
DRAW 255, 175
```

Podemos observar que 255 es el incremento máximo que podemos poner en horizontal, ya que coincide con la dimensión de la pantalla. Con 175 sucede lo mismo y es, por tanto, el máximo incremento en vertical. Análogamente, en el Amstrad tendremos que poner los parámetros que coincidan con las dimensiones máximas:

```
DRAWR 639,399
```

Cualquiera de los dos comandos anteriores nos darán un resultado en pantalla como el de la figura 7.2.

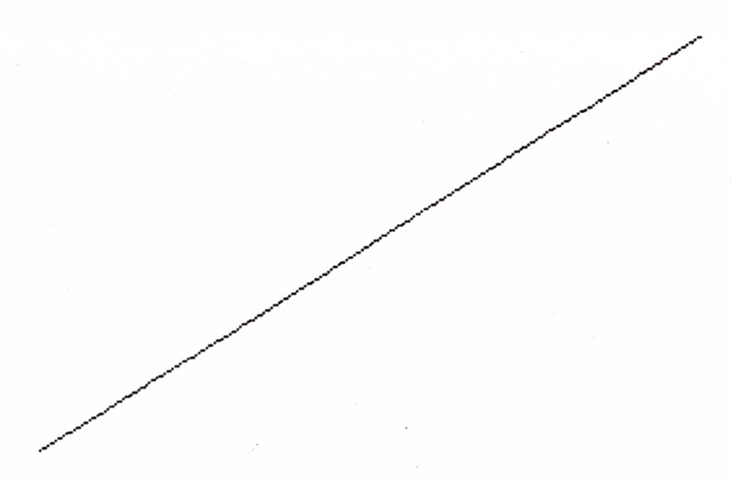


Fig. 7.2. Trazado de una diagonal en pantalla.

Después de trazar una recta, el cursor de gráficos quedará situado en el extremo final de dicha recta, es decir, que en nuestros ejemplos anteriores, después de trazar la diagonal, el cursor gráfico estará en el punto 255,175, en el caso del Spectrum, y en el punto 639,399, si estamos trabajando en el Amstrad. Esto significa que si ahora queremos trazar otra recta a continuación de la anterior no tenemos más que teclear otro DRAW (Spectrum) o DRAWR (Amstrad) con sus correspondientes parámetros de incremento. Pero tenemos que fijarnos en el detalle siguiente: el cursor está situado en el ángulo superior derecho, lo que significa que no podemos trazar otra recta que continúe avanzando hacia la derecha o hacia arriba, ya que se saldría de la pantalla. La nueva recta tendrá que retroceder o descender para que se sitúe dentro de los límites. Para indicarle al ordenador que la recta tiene que retroceder o descender basta con poner, en los parámetros de incrementos, números negativos. Probemos con el programa siguiente, para el Spectrum:

```
10 DRAW 255,175
20 DRAW -128,-175
```

Un programa análogo para el Amstrad sería:

```
10 DRAW 639,399
20 DRAWR -320, -399
```

En ambos casos podemos comprobar cómo la línea 20 del programa se encarga de trazar una recta que retrocede y desciende. El resultado en pantalla está representado en la figura 7.3.

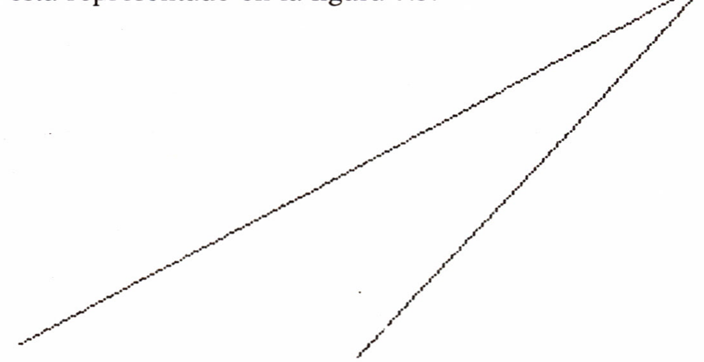


Fig. 7.3. Trazado de dos rectas consecutivas en pantalla.

Visto todo lo anterior, deducimos que, si queremos trazar una recta horizontal, el incremento vertical (incremento y) será cero, mientras que si trazamos una recta vertical el incremento horizontal (incremento x) también será cero.

RECTAS	Δx	Δy
↗	+	+
↘	+	−
↙	−	−
↖	−	+
↑	Ø	+
→	+	Ø
↓	Ø	−
←	−	Ø

Fig. 7.4. Tabla-resumen de DRAW (Spectrum) y DRAWR (Amstrad).

Por otra parte, si queremos trazar rectas independientes, es decir, que no vayan una a continuación de otra, tendremos que trasladar el cursor de gráficos al punto donde queremos empezar. Ya vimos en el capítulo anterior cómo hacer esto. Recordemos que PLOT dibuja un punto en la pantalla de alta resolución y deja el cursor gráfico en ese punto. Además, el Amstrad cuenta con la instrucción MOVE, que mueve el cursor gráfico al punto de la pantalla que indiquemos, pero sin imprimir dicho punto. Probemos ahora a trazar las dos diagonales de la pantalla.

```
10 REM*DIAGONALES DEL SPECTRUM*
20 DRAW 255,175
30 PLOT 0,175
40 DRAW 255, -175

10 REM *DIAGONALES DEL AMSTRAD*
20 DRAWR 639, 399
30 MOVE 0, 399
40 DRAWR 639, -399
```

El resultado de la ejecución de cualquiera de estos dos programas nos dará como resultado dos rectas independientes (no una a continuación de otra), como se muestra en la figura 7.5

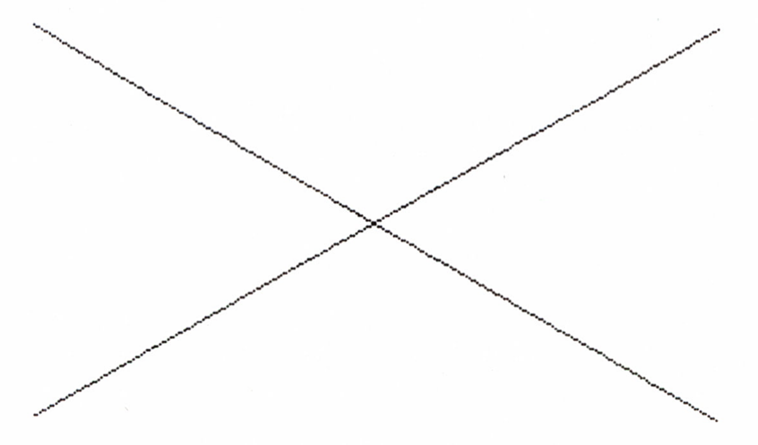


Fig. 7.5. Trazado de las dos diagonales de la pantalla.

El Amstrad dispone de la instrucción DRAW, además de la DRAWR ya vista, para el trazado de rectas, aunque no funciona como el DRAW del Spectrum. La instrucción DRAW tiene en el Amstrad el formato siguiente:

N. de línea DRAW x,y

donde x e y son las coordenadas del punto de la pantalla donde queremos que finalice la recta. Por tanto, la diferencia entre DRAW y DRAWR es que el primero utiliza coordenadas absolutas, es decir, referidas siempre al origen de coordenadas, mientras que DRAWR utiliza coordenadas relativas, es decir, referidas al último punto trazado. Podemos teclear los dos siguientes programas para ver la diferencia:

```
10 DRAW 200, 200
20 DRAW 440, 200
30 DRAW 640, 400

10 DRAWR 200, 200
20 DRAWR 240, 0
30 DRAWR 200, 200
```

Como podemos ver, los dos programas anteriores usan distintos parámetros para las coordenadas; sin embargo, si los ejecutamos obtendremos el mismo resultado.

Evidentemente, con la instrucción DRAW del Amstrad no podemos usar parámetros negativos, ya que estos puntos estarían situados fuera de la pantalla, como se muestra en la figura 7.6.

Sin embargo, esto sólo sucede cuando el origen está situado en el punto 0,0 (ángulo inferior izquierdo). Si trasladamos el origen, mediante la instrucción ORIGIN, a otro punto de la pantalla, entonces sí podremos utilizar parámetros negativos en el DRAW.

El BASIC Simon del Commodore dispone de la instrucción LINE, para el trazado de rectas, y tiene el siguiente formato:

```
LINE X1, Y1, X2, Y2, T
```

donde X1, Y1 son las coordenadas del punto de inicio de la recta y X2,Y2 son las coordenadas del punto donde finaliza la recta. El parámetro T in-

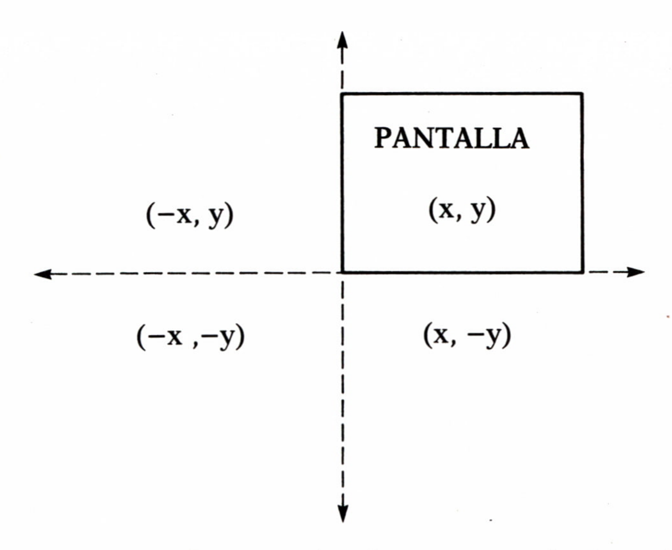


Fig. 7.6. Los puntos con alguna coordenada negativa quedan fuera de la pantalla con ORIGIN 0,0. (Amstrad)

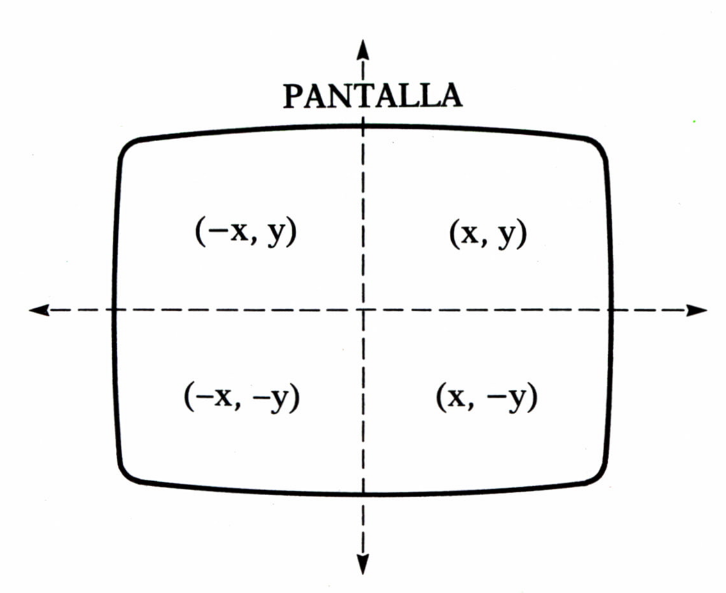


Fig. 7.7. Trasladando el origen a otro punto de la pantalla, como, por ejemplo, el centro, tendremos parámetros positivos y negativos para la instrucción DRAW (Amstrad)

dica si los puntos que forman dicha recta están encendidos (1) o apagados (0).

Bueno, pues ya sabemos dibujar todo tipo de rectas, así que ya podemos pasar a las circunferencias.

Para dibujar circunferencias en el Spectrum contamos con la instrucción CIRCLE, que nos permite trazar circunferencias de cualquier tamaño. El formato es el siguiente:

N. de línea CIRCLE x,y,radio

donde x e y son las coordenadas del centro de la circunferencia y radio indica el radio de la circunferencia.

Si introducimos el siguiente comando:

```
CIRCLE 128, 88, 87
```

obtendremos la circunferencia más grande que podemos dibujar en la pantalla del Spectrum. Las coordenadas elegidas para el centro de la circunferencia coinciden con el centro de la pantalla. El radio es el máximo que podemos dar, puesto que si lo damos mayor obtendremos un mensaje de error, ya que la circunferencia se saldría de la pantalla.

```
  10 REM ***********************
  20 REM * CIRCUNFERENCIAS EN *
  30 REM *      SPECTRUM       *
  40 REM ***********************
  50 INPUT "COORDENADAS DEL CENT
RO";X,Y
  60 IF X>255 OR Y>175 THEN GO T
O 40
  70 INPUT "RADIO";R
  80 IF R>X OR R>255-X OR R>Y OR
 R>175-Y THEN GO TO 70
  90 CIRCLE X,Y,R
```

El programa 7.2 nos permite dibujar todo tipo de circunferencias en cualquier parte de la pantalla. Podemos ver un ejemplo en la figura 7.8.

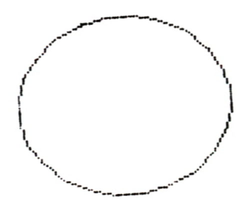


Fig. 7.8. Circunferencia trazada con la instrucción CIRCLE del Spectrum.

El Amstrad no dispone de instrucción para el trazado de circunferencias; sin embargo, esto no significa que no podamos dibujarlas. Mediante la instrucción **DRAW** y con unas nociones básicas de trigonometría podemos trazar circunferencias en cualquier zona de la pantalla y de cualquier tamaño.

```
10 REM ******************
20 REM * CIRCUNFERENCIAS *
30 REM *  PARA AMSTRAD    *
40 REM ******************
50 CLS
60 INPUT "COORDENADAS DEL CENTRO";X,Y
70 IF X>639 OR X<0 OR Y>399 OR Y<0
   THEN GOTO 60
80 INPUT "RADIO";R
90 IF R>X OR R>639-X OR R>Y OR
   R>399-Y THEN GOTO 80
100 CLS:CLG
110 ORIGIN X,Y
120 MOVE R,0
130 FOR A=0 TO 2*PI STEP PI/180
140 DRAW R*COS(A),R*SIN(A)
150 NEXT
```

El programa 7.3 nos permite trazar todo tipo de circunferencias. Para ello sitúa el origen de coordenadas en el centro de la circunferencia y mueve el cursor gráfico hasta el punto de dicha circunferencia situado más a la derecha. Después, basándose en la idea de que una circunferencia no es más que un polígono de muchos lados, traza un polígono de 360 lados, haciendo que cada lado vaya girando un grado respecto al origen. La figura 7.8 muestra esta idea de forma esquemática.

El bucle de la línea 130 se encarga de recorrer los 360 grados de la circunferencia (2 radianes), mientras que la instrucción **DRAW** de la línea siguiente se encarga de trazar los 360 lados que van a formar la circunferencia.

Otro método para trazar circunferencias es el utilizado en el programa 7.4. Este método también se basa en propiedades trigonométricas y en la idea del círculo como polígono de muchos lados. La ejecución de este programa es más rápida que la del programa 7.3, ya que sólo tiene que calcular el seno y el coseno una vez (en la línea 130).

```
10 REM ************************
20 REM *   CIRCUNFERENCIAS     *
30 REM *      EN AMSTRAD       *
40 REM ************************
```

```
50 CLS
60 INPUT "COORDENADAS DEL CENTRO";X,Y
70 IF X>639 OR X<0 OR Y>399 OR Y<0
   THEN GOTO 60
80 INPUT "RADIO";R
90 IF R>X OR R>639-X OR R>Y OR
   R>399-Y OR R<1 THEN 80
100 CLS:CLG
110 ORIGIN X,Y
120 MOVE 0,R
130 ca=COS(PI/90):sa=SIN(PI/90)
140 x1=0:Y1=R
150 FOR i=0 TO 360 STEP 2
160 x2=x1*ca-Y1*SA
170 Y2=Y1*ca+x1*sa
180 DRAW x2,y2
190 x1=x2:y1=y2
200 NEXT
```

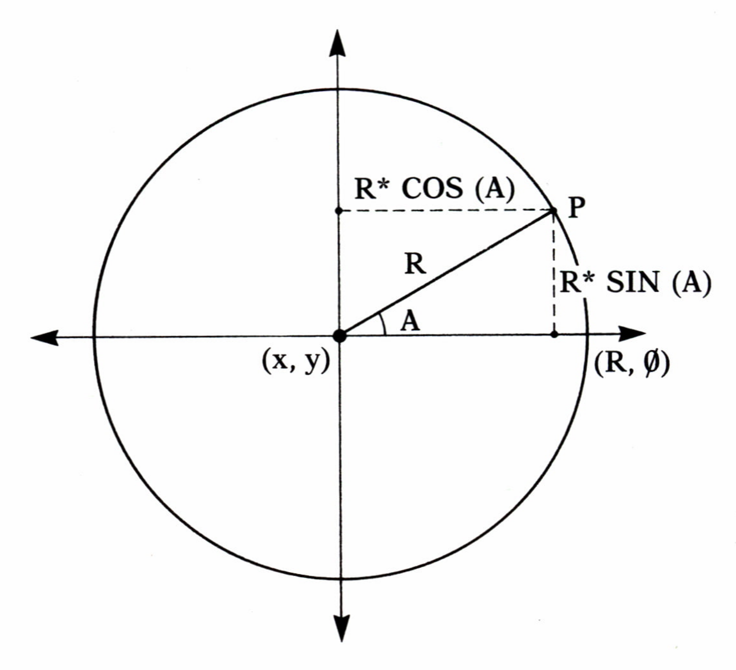


Fig. 7.9. Esquema para el trazado de circunferencias, en Amstrad.

Esto de trazar circunferencias está muy bien, pero ¿qué pasa si sólo queremos trazar un arco de circunferencia? En el caso del Spectrum resulta bastante sencillo, ya que la instrucción DRAW nos permite dibujar, ade-

más de rectas como ya hemos visto, arcos de circunferencia. El formato de DRAW para trazar arcos es el siguiente:

N. de línea **DRAW** incremento x, incremento y, ángulo

donde *incremento x* e *incremento y* determinan el punto donde va a finalizar el trazado del arco, referido al punto donde se inicia el trazado, y *ángulo* indica el ángulo en radianes que va a abarcar el arco de circunferencia.

Hay que tener en cuenta que los ángulos se miden en sentido antihorario; por tanto, tenemos que cuidar los signos dependiendo del lado por el que queramos trazar el arco.

```
10 REM ***********************
20 REM * ARCOS CIRCUNFERENCIA*
30 REM *       SPECTRUM      *
40 REM ***********************
50 DRAW 0,175,PI
60 DRAW 255,0,PI
70 DRAW 0,-175,PI
30 DRAW -255,0,PI
```

En el programa 7.5 utiliza siempre valores positivos del ángulo y, en cambio, va variando los signos de los incrementos de x e y, de modo que el trazado de cada semicírculo sigue siempre un sentido antihorario (signo positivo en el ángulo).

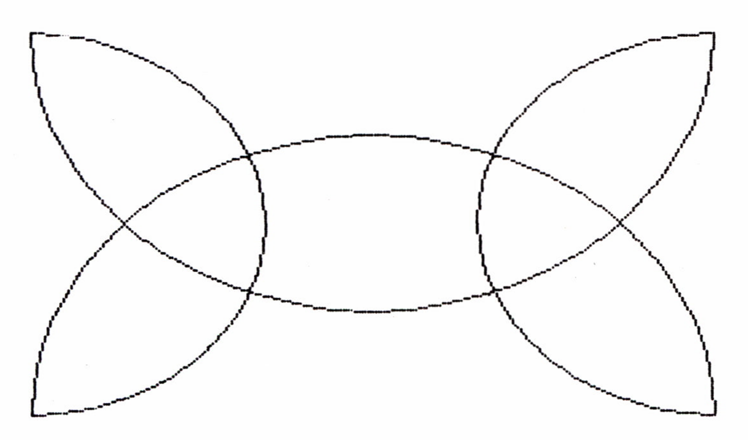


Fig. 7.10. Resultado de la ejecución del programa 7.5.

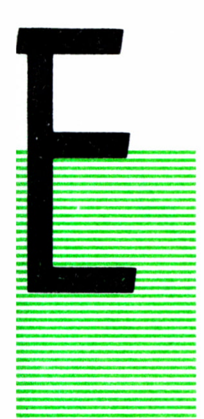
En el Amstrad el trazado de arcos de circunferencia se consigue con un programa similar al de trazado de circunferencias completas.

El programa 7.6 traza todo tipo de arcos de circunferencia, basándose en el programa 7.3.

```
10 REM ****************************
20 REM * ARCOS DE CIRCUNFERENCIA *
30 REM *         AMSTRAD          *
40 REM ****************************
50 CLS
60 INPUT "COORDENADAS DEL CENTRO ";XC,YC
70 IF XC>639 OR XC<0 OR YC>399 OR YC<0
   THEN GOTO 60
80 INPUT "RADIO ";R
90 IF R>XC OR R>639-XC OR R>YC
   OR R>399-YC OR R<1 THEN GOTO 80
100 INPUT "ANGULOS INICIAL Y FINAL RESPECTO
    AL EJE HORIZONTAL (GRADOS) ";AI,AF
110 IF AI<0 OR AF<0 OR AF>360
    OR AF <AI THEN GOTO 100
120 AI=AI*PI/180
130 AF=AF*PI/180
140 CLS:CLG
150 ORIGIN XC,YC
160 MOVE R*COS(AI),R*SIN(AI)
170 FOR A=AI TO AF STEP PI/180
180 DRAW R*COS(A),R*SIN(A)
190 NEXT
```

Bueno, pues ya sabemos trazar toda clase de puntos, rectas, circunferencias y arcos, así que ya podemos pasar a dibujar figuras diversas componiendo todos estos elementos.

FIGURAS GEOMETRICAS Y ARTE ABSTRACTO 8



EN este capítulo vamos a ver algunos ejemplos de programas para trazar distintos tipos de figuras geométricas. Si además queremos variar el color de la tinta y del papel, no tenemos más que añadir las instrucciones sobre colores vistas en el capítulo 3, ya que los colores en alta resolución son los mismos que en baja resolución. Sólo hay un pequeño matiz en el Spectrum y es que el color de la tinta no se puede definir para un solo pixel, sino para todo el carácter que contiene dicho pixel. Esto significa que no podemos utilizar distintos colores para los pixels contenidos en un mismo carácter del Spectrum.

En los programas 8.3 y 8.4 podemos ver cómo variar los colores de tinta y papel en el trazado de estrellas. Esto es extensible a cualquier otro programa.



POLIGONOS REGULARES

Las figuras geométricas más sencillas que podemos trazar son los polígonos regulares cerrados, es decir, triángulos equiláteros, cuadrados, pentágonos, etc.

Podríamos diseñar un programa para trazar cada una de estas figuras, pero resulta mucho más práctico que un solo programa pueda dibujar cualquier polígono regular que nosotros le indiquemos. Este es el objetivo del programa 8.1 para el Spectrum.

```
10 REM ************************
20 REM * POLIGONOS REGULARES  *
30 REM *        SPECTRUM      *
40 REM ************************
```

```
  60 INPUT "NUMERO DE LADOS ";L
  70 LET L=INT L
  80 IF L<3 OR L>50 THEN GO TO 5
0
  90 INPUT "RADIO ";R
 100 IF R>86 OR R<1 THEN GO TO 9
0
 110 INPUT "ANGULO DEL PRIMER RA
DIO CON EL  EJE HORIZONTAL (GRAD
OS) ";A
 120 IF A<0 OR A>360 THEN GO TO
110
 130 LET A=A*PI/180
 140 LET P=2*PI/L
 150 LET X1=127+R*COS A: LET Y1=
87+R*SIN A
 160 PLOT X1,Y1
 170 FOR I=A+P TO A+2*PI+0.02 ST
EP P
 180 LET X2=127+R*COS I: LET Y2=
87+R*SIN I
 190 DRAW X2-X1,Y2-Y1
 200 LET X1=X2: LET Y1=Y2
 210 NEXT I
```

Al ejecutar el programa 8.1 tendremos que dar los datos siguientes: número de lados del polígono, radio (recordemos que todos los polígonos regulares son inscribibles en una circunferencia) y ángulo que forma el primer vértice con la horizontal (esto nos permite dibujar el polígono en cualquier posición). Hemos situado el centro de los polígonos coincidiendo con el centro de la pantalla.

Introduciendo los datos que deseemos obtendremos todo tipo de polígonos, como, por ejemplo, el octógono de la figura 8.1.

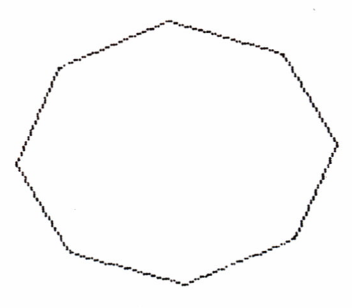


Fig. 8.1. Octágono trazado con el programa para dibujar polígonos regulares.

Si al programa 8.1 le añadimos al final la línea:

```
220 GOTO 60
```

tendremos la posibilidad de introducir datos diferentes, consiguiendo así infinidad de figuras compuestas por polígonos regulares, como las que se muestran en la figura 8.2.

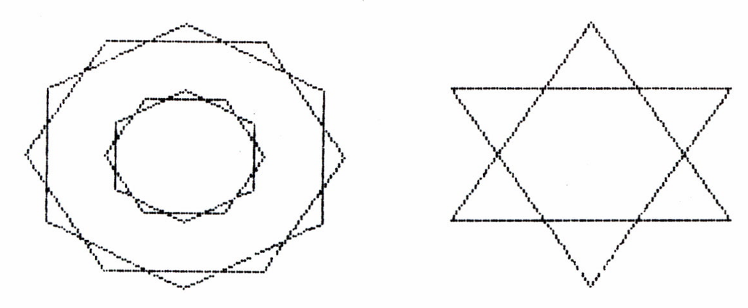


Fig. 8.2. Figuras diversas trazadas combinando varios polígonos.

La versión de este programa de trazado de polígonos para el Amstrad es la presentada en el programa 8.2.

```
10 REM ***********************
20 REM * POLIGONOS REGULARES *
30 REM *       AMSTRAD       *
40 REM ***********************
50 CLS
60 INPUT "NUMERO DE LADOS ";L
65 L=CINT(L)
70 IF L<3  THEN GOTO 60
80 INPUT "RADIO ";R
90 IF R>199 OR R<1 THEN GOTO 80
100 INPUT "ANGULO DEL PRIMER RADIO
 CON EL EJE HORIZONTAL (GRADOS) ";A
110 IF A<0 OR A>360 THEN GOTO 100
120 CLS
130 A=A*PI/180
140 P=2*PI/L
150 ORIGIN 320,200
160 MOVE R*COS(A),R*SIN(A)
170 FOR I=A+P TO A+2*PI+0.02 STEP P
180 DRAW R*COS(I),R*SIN(I)
190 X1=X2:Y1=Y2
200 NEXT I
```

ESTRELLAS REGULARES

Para el dibujo de estrellas nos vamos a basar en los mismos principios utilizados para el trazado de polígonos, es decir, tenemos que manejar las mismas leyes trigonométricas (la coordenada en X es R cosA y la coordenada en Y es R.senA), sólo que esta vez las aplicamos para dos circunferencias diferentes (la mayor abarca las puntas exteriores y la menor las interiores).

El programa 8.3 nos permite dibujar todo tipo de estrellas en el centro de la pantalla del Spectrum.

```
  10 REM *************
  20 REM * ESTRELLAS *
  30 REM * SPECTRUM  *
  40 REM *************
  45 INPUT "COLORES DE TINTA Y P
APEL";TI,PA
  50 IF TI>7 OR TI<0 OR PA>7 OR
PA<0 THEN GO TO 42
  60 INK TI: PAPER PA: CLS
  70 INPUT "NUMERO DE PUNTAS";N
  80 IF N<=2 THEN GO TO 50
  90 INPUT "RADIOS MAYOR Y MENOR
";R1,R2
 100 IF R1>86 OR R1<1 OR R2<1 OR
 R1<R2 THEN GO TO 90
 110 INPUT "ANGULO DE LA PRIMERA
 PUNTA CON  EL EJE HORIZONTAL (G
RADOS)";A
 120 IF A<0 OR A>360 THEN GO TO
110
 130 LET A=A*PI/180
 140 LET P=PI/N
 150>LET X1=127+R1*COS A: LET Y1
=87+R1*SIN A
 160 PLOT X1,Y1
 170 FOR I=1 TO N
 180 LET A=A+P
 190 LET X2=127+R2*COS A: LET Y2
=87+R2*SIN A
 200 DRAW X2-X1,Y2-Y1
 210 LET A=A+P
 220 LET X1=X2: LET Y1=Y2
 230 LET X2=127+R1*COS A: LET Y2
=87+R1*SIN A
```

```
240 DRAW X2-X1,Y2-Y1
250 LET X1=X2: LET Y1=Y2
260 NEXT I
```

Un ejemplo de la ejecución del programa 8.3 lo tenemos en la figura 8.3, que representa una estrella de 12 puntas.

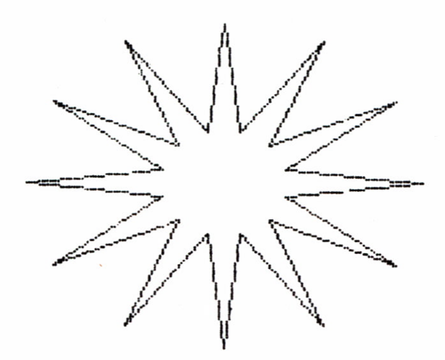


Fig. 8.3. Estrella obtenida con el programa para trazado de estrellas.

Si queremos dibujar estrellas en el Amstrad utilizaremos el programa 8.4, que es análogo al 8.3.

```
10 REM *************
20 REM * ESTRELLAS *
30 REM *  AMSTRAD  *
40 REM *************
50 CLS
60 INPUT "COLORES DE TINTA Y PAPEL ";
   TI,PA
70 IF TI>26 OR TI<0 OR PA>26 OR PA<0
   THEN 60
80 INK 1,TI:INK 0,PA:CLS
90 INPUT "NUMERO DE PUNTAS ";N
100 IF N<=2 THEN 90
110 INPUT "RADIOS MAYOR Y MENOR ";R1,R2
120 IF R1>199 OR R1<1 OR R2<1 OR R1<R2
    THEN 110
130 INPUT "ANGULO DE LA PRIMERA PUNTA
    CON EL EJE HORIZONTAL (GRADOS) ";A
140 IF A<0 OR A>360 THEN 130
150 CLS:CLG
```

```
160 A=A*PI/180
170 P=PI/N
180 X1=R1*COS(A)  :Y1=R1*SIN(A)
190 ORIGIN 320,200
200 MOVE X1,Y1
210 FOR I=1 TO N
220 A=A+P
230 X2=R2*COS(A):Y2=R2*SIN(A)
240 DRAW X2,Y2
250 A=A+P
260 X1=X2:Y1=Y2
270 X2=R1*COS(A):Y2=R1*SIN(A)
280 DRAW X2,Y2
290 X1=X2:Y1=Y2
300 NEXT I
```

Podemos observar que los programas para trazados de circunferencias, polígonos, estrellas y, en general, cualquier figura inscribible en una circunferencia, tienen todos una cierta similitud, ya que todos se basan en las propiedades trigonométricas de la circunferencia y en la repetición de dichas propiedades para distintos ángulos. Combinando estas características podemos trazar figuras más complejas a partir de la combinación de figuras más sencillas.



FLORES DE CIRCUNFERENCIAS

Un ejemplo de lo expuesto anteriormente son las conocidas flores de circunferencias. La idea del programa consiste en tomar una circunferencia como pétalo e ir trasladando su centro a lo largo de otra circunferencia principal que no se dibuja. Esta idea se refleja de forma esquemática en la figura 8.4.

Pues bien, el programa 8.5 sigue esta idea y nos permite trazar todo tipo de flores de circunferencias en el centro de la pantalla del Spectrum.

Recordemos que en el Spectrum disponemos de la instrucción CIRCLE, que en este programa nos permite el trazado de los pétalos. Si queremos desarrollar este programa en el Amstrad tendremos que utilizar la rutina para el trazado de circunferencias (programas 7.3 ó 7.4). En la figura 8.5 podemos ver algunos resultados obtenidos con el programa 8.5.

Como hemos visto, las posibilidades de este tipo de programas son infinitas. Podríamos dibujar flores de polígonos, flores de estrellas, combinar ambas figuras o unir todos los vértices de un polígono entre sí, dando

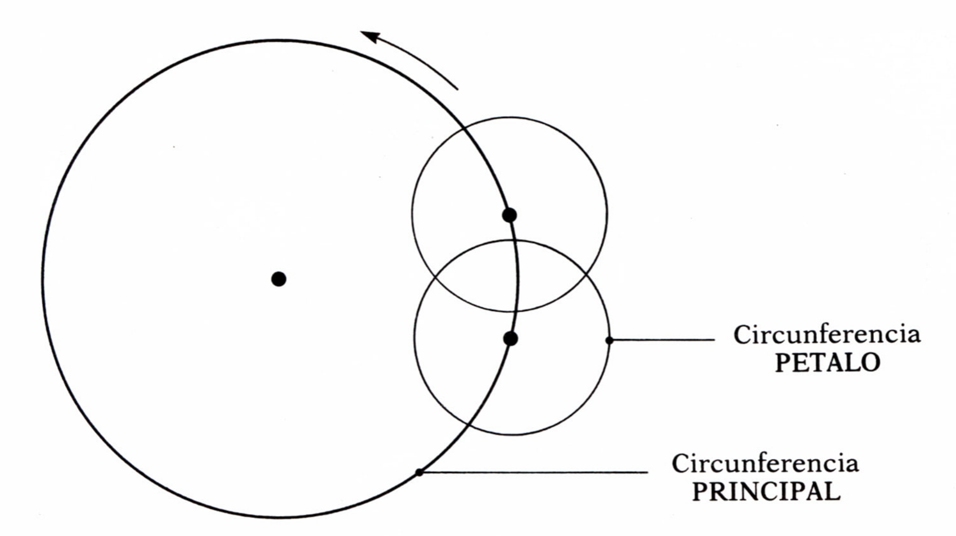


Fig. 8.4. Esquema para el trazado de flores de circunferencia.

lugar a los dibujos conocidos como «clavos y cuerdas». Pero éstas son sólo algunas ideas que el lector puede trasladar a programas junto con todas aquellas que pueda imaginar, siguiendo esquemas de rutinas similares a los vistos en los ejemplos.

```
  10 REM ********************
  20 REM *      FLOR DE      *
  30 REM * CIRCUNFERENCIAS   *
  40 REM *      SPECTRUM     *
  50 REM ********************
  60 INPUT "NUMERO DE PETALOS":N
  70 IF N<2 THEN GO TO 60
  80 INPUT "RADIO PRINCIPAL";R1
  90 IF R1>86 OR R1<1 THEN GO TO
 80
 100 INPUT "RADIO DE LOS PETALOS
";R2
 110 IF R2>86-R1 OR R2<1 THEN GO
 TO 100
 120 LET P=2*PI/N
 130 LET A=0
 140 FOR I=1 TO N
 150 LET X=127+R1*COS A: LET Y=8
7+R1*SIN A
 160 CIRCLE X,Y,R2
 170 LET A=A+P
 180 NEXT I
```

Fig. 8.5. Ejemplos de flores de circunferencia obtenidas con el programa 8.5.



FIGURAS ALEATORIAS Y DIBUJOS ABSTRACTOS

Gracias al empleo del ordenador y con unos cuantos programas sencillos podemos convertirnos en unos grandes artistas de temas abstractos.

El programa 8.6 realiza dibujos abstractos en el Spectrum a base de rectas de diferentes longitudes, direcciones y colores. Esto se consigue haciendo que el ordenador elija un color al azar para el papel de fondo y a continuación, mediante un bucle infinito, va eligiendo sucesivamente y de forma aleatoria color, punto de inicio y punto de final de cada recta. Cuando tengamos en pantalla un «cuadro» que nos guste no tenemos más que hacer un BREAK para detener la ejecución.

```
 10 REM ***********************
 20 REM * DIBUJOS ABSTRACTOS *
 30 REM ***********************
 40 LET P=INT (RND*8)
 50 PAPER P: CLS
 60 LET T=INT (RND*8)
 70 INK T
 80 LET X=INT (RND*256)
 90 LET Y=INT (RND*176)
100 PLOT X,Y
110 LET X1=INT (RND*256)
120 LET Y1=INT (RND*176)
130 DRAW X1-X,Y1-Y
140 GO TO 60
```

La figura 8.6 es una muestra (en blanco y negro) del tipo de dibujos que podemos obtener. Si utilizamos una pantalla de TV en color conseguiremos unos bonitos «cuadros» llenos de colorido y fantasía.

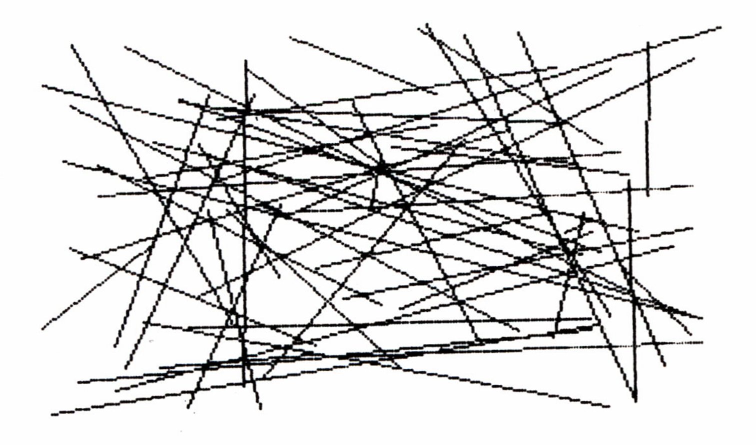
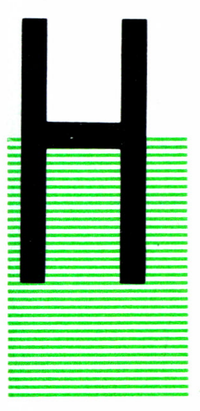


Fig. 8.6. Cuadro abstracto pintado con el programa 8.6.

El programa 8.7 realiza también «cuadros» abstractos en el Amstrad, pero esta vez utilizando circunferencias de diferentes tamaños y colores. La idea es la misma que para la realización del programa anterior, es decir, el ordenador elige al azar el centro de la circunferencia, el radio y el color. Los resultados pueden ser sorprendentes.

```
10 REM ************************
20 REM * CIRCULOS ALEATORIOS *
30 REM ************************
40 MODE 0
50 CLS :CLG
60 c1=INT(RND(1)*640+100)
70 c2=INT(RND(1)*400+100)
80 r=INT(RND(1)*60+10)
90 co=INT(RND(1)*14)
100 GOSUB 120
110 GOTO 60
120 ORIGIN c1,c2
130 MOVE 0,r
140 ca=COS(PI/45):sa=SIN(PI/45)
150 x1=0:y1=r
160 FOR i = 0 TO 360 STEP 4
170 x2=x1*ca-y1*sa
180 y2=y1*ca+x1*sa
190 DRAW x2,y2,co
200 x1=x2:y1=y2
210 NEXT
220 RETURN
```

COLOREANDO 9



ASTA ahora las figuras que hemos realizado han sido figuras a base de líneas de colores diversos, pero no hemos podido hacer figuras sólidas, esto es, figuras rellenas de color. Por ello, en este capítulo vamos a aprender la forma de colorear desde un programa a una figura cerrada, realizada mediante líneas de color. Sin embargo, hay que advertir que las rutinas de rellenado de figuras son muy lentas, y la única forma de que fueran verdaderamente útiles sería realizarlas en Código Máquina, lo que se sale del ámbito de este libro. Por eso, es conveniente restringir su uso a figuras de tamaño pequeño para no eternizarnos delante de la pantalla del ordenador.



LA FUNCION POINT

Antes de empezar con las rutinas de coloreado, hay que explicar una función que vamos a necesitar para poder realizarlas.

Todas las rutinas de coloreado se basan en ir mirando los puntos de alrededor de uno dado y comprobar si están pintados o no. Pues bien, para esto existe en el ordenador una función que nos permite averiguar si un punto determinado de la pantalla está o no pintado. En el caso del Spectrum, la función se llama **POINT**, y en el Amstrad, se llama **TEST**.

La función **POINT** necesita dos argumentos: la coordenada X y la coordenada Y del punto que queremos mirar. Esta función devuelve el valor 1 si el punto está pintado y el valor 0 en caso contrario.

La función **TEST** funciona de forma parecida, pero devuelve el valor de la tinta usada en esa posición de la pantalla.

También existe en el Amstrad la función **TESTR**, que en vez de necesitar las coordenadas del punto como argumento, necesita dos números que son desplazamientos relativos al punto actual.

```
10 REM ************************
20 REM * PRUEBA DE LA FUNCION*
30 REM *        POINT X,Y        *
40 REM ************************
50 CLS
60 FOR I=1 TO 1000
70 PLOT INT (RND*256),INT (RND
*176)
80 NEXT I
90 GO TO 80
```

```
10 REM ************************************
20 REM *   PRUEBA DE LA FUNCION TEST X,Y    *
30 REM ************************************
40 CLS:CLG
50 FOR I=1 TO 1000
60 PLOT RND(1)*640,RND(1)*400
70 NEXT I
80 LOCATE 1,25:PRINT "EL VALOR DEL PUNTO ES :"
   ;TEST (RND(1)*640,RND(1)*400)
90 GOTO 80
```

Estos programas simplemente llenan la pantalla de mil puntos distribuidos aleatoriamente por ella, y luego van comprobando aleatoriamente puntos de la pantalla con la función **POINT** o **TEST** y poniendo su resultado en la pantalla.



COLOREANDO FIGURAS

Antes de explicar la rutina de coloreado general, vamos a realizar un programa para ver cómo se puede colorear la figura resuelta.

La idea es fácil de entender: para cada línea de la pantalla buscar el punto del borde del polígono empezando por la izquierda. Buscar, a continuación, el punto del borde derecho del polígono en esa misma línea y unir los dos puntos con una línea de color. Si repetimos esto para todas las líneas de la pantalla tendremos los siguientes programas:

```
10 REM ***********************
20 REM * PROG. DE COLOREADO *
30 REM ***********************
```

```
   40 CLS : CIRCLE 127,87,85
   50 FOR Y=0 TO 175
   60 LET X1=0
   70 IF POINT (X1,Y1)=1 THEN
    GO TO 100
   80 LET X1=X1+1
   90 IF X1<=255 THEN GO TO 70
  100 LET XR=255
  110 IF POINT (XR,Y)=1 THEN GO T
O 140
  120 LET XR=XR-1
  130 IF XR>=0 THEN GO TO 110
  140 IF X1>=256 THEN GO TO 160
  150 PLOT X1,Y: DRAW XR-X1,0
  160 NEXT Y
```

```
10 REM ***************************
20 REM *   PROG. DE COLOREADO    *
30 REM ***************************
40 MODE 1:CLG
50 ORIGIN 320,200:MOVE 0,199
60 CA=COS(PI/90):SA=SIN(PI/90)
70 X1=0:Y1=199
80 FOR I=0 TO 360 STEP 2
90 X2=X1*CA-Y1*SA
100 Y2=Y1*CA+X1*SA
110 DRAW X2,Y2
120 X1=X2:Y1=Y2
130 NEXT
135 ORIGIN 0,0
140 FOR Y=0 TO 400 STEP 2
150 X1=0
160 IF TEST(X1,Y) >0 THEN GOTO 190
170 X1=X1+2
180 IF X1<=639 THEN GOTO 160
190 XR=639
200 IF TEST(XR,Y)>0 THEN GOTO 230
210 XR=XR-2
220 IF XR>=0 THEN GOTO 200
230 IF X1 >= 640 THEN GOTO 250
240 DRAWR X1-XR,0
250 NEXT Y
```

Como vemos, el programa es relativamente lento. Además, esta rutina sólo vale para figuras convexas, esto es, «sin picos hacia dentro» y sólo sirve para dibujos con una figura.

Vamos a ver un ejemplo de mal funcionamiento de esta rutina.

```
   10 REM ************************
   20 REM * PROG. DE COLOREADO 2*
   30 REM ************************
   40 CLS : PLOT 100,50: DRAW 50,
-50: DRAW -50,100: DRAW -50,-100
: DRAW 50,50
   50 FOR Y=0 TO 175
   60 LET X1=0
   70 IF POINT (X1,Y1)=1 THEN
    GO TO 100
   80 LET X1=X1+1
   90 IF X1<=255 THEN GO TO 70
  100 LET XR=255
  110 IF POINT (XR,Y)=1 THEN GO T
O 140
  120 LET XR=XR-1
  130 IF XR>=0 THEN GO TO 110
  140 IF X1>=256 THEN GO TO 160
  150 PLOT X1,Y: DRAW XR-X1,0
  160 NEXT Y
```

```
10 REM ***************************
20 REM *   PROG. DE COLOREADO 2 *
30 REM ***************************
40 MODE 1:CLG
50 PLOT 100,50:DRAWR 50,-50:DRAWR -50,100
   :DRAWR -50,-100:DRAWR 50,50
135 ORIGIN 0,0
140 FOR Y=0 TO 400 STEP 2
150 X1=0
160 IF TEST(X1,Y) >0 THEN GOTO 190
170 X1=X1+2
180 IF X1<=639 THEN GOTO 160
190 XR=639
200 IF TEST(XR,Y)>0 THEN GOTO 230
210 XR=XR-2
220 IF XR>=0 THEN GOTO 200
230 IF X1 >= 640 THEN GOTO 250
240 DRAWR X1-XR,0
250 NEXT Y
```

No colorea la figura adecuadamente, pinta «de más», ya que las líneas interiores del «hueco» no las detecta.

Por tanto, hay que buscar otra forma de colorear figuras que sea más general y sirva para todo tipo de figuras cerradas.

El método de rellenado general que vamos a diseñar se basa en el siguiente principio:

Cogemos al azar un punto *interior* de la figura y lo coloreamos. A partir de ese punto vamos mirando los 4 puntos que están encima, debajo, a la izquierda, a la derecha, y si están sin pintar, los pintamos y les aplicamos el mismo procedimiento.

Evidentemente el procedimiento terminará cuando todos los puntos que comprobemos estén ya pintados, lo que significará que hemos llegado al borde de la figura.

La rutina de rellenado es la siguiente:

```
  35 DIM p(500,2)
  37 DEF FN z(v)=INT ((((v+1)/50
0)-(INT ((v+1)/500)))*500)+1
 100 LET pr=1: LET ul=0
 110 LET xa=x1: LET ya=y1: GO SU
B 200
 120 LET x=p(pr,1): LET y=p(pr,2
): LET pr=FN z(pr)
 130 LET xa=x: LET ya=y+1: GO SU
B 200
 140 LET xa=x: LET ya=y-1: GO SU
B 200
 150 LET xa=x+1: LET ya=y: GO SU
B 200
 160 LET xa=x-1: LET ya=y: GO SU
B 200
 170 IF pr<>FN z(ul) THEN GO TO
120
 180 RETURN
 200 IF POINT (xa,ya)>0 THEN RET
URN
 210 PLOT xa,ya
 220>LET ul=FN z(ul)
 230 LET p(ul,1)=xa: LET p(ul,2)
=ya
 240 RETURN
```

La matriz P es una «cola», que es una forma de guardar los datos, de tal forma que el primer dato que guardaremos será el primero que saquemos. Por ejemplo, si almacenamos por este orden los numeros 4,5,7,8 y 9, después, al sacar un dato , el primero que saquemos será el 4, luego el 5, 7, y así sucesivamente. El nombre de «cola» es porque se comporta exactamente igual que una cola de personas, el primero que llega es el primero en ser atendido. Esta matriz nos va a servir para ir guardando los puntos que tenemos que comprobar.

```
100 PR=1:UL=0
110 XA=X1:YA=Y1:GOSUB 200
120 X=P(PR,1):Y=P(PR,2): pr=(pr+1) MOD 500
130 XA=X:YA=Y+2:GOSUB 200
140 XA=X:YA=Y-2:GOSUB 200
150 XA=X+2:YA=Y:GOSUB 200
160 XA=X-2:YA=Y:GOSUB 200
170 IF PR<>((ul+1) MOD 500) THEN GOTO 120
180 RETURN
200 IF TEST (XA,YA)>0 THEN RETURN
210 PLOT XA,YA
220 UL= (ul+1) MOD 500
230 P(UL,1)=XA:P(UL,2)=YA
240 RETURN
```

El programa funciona de la siguiente forma: en la línea 100 inicializamos el principio y el final de la «cola» para saber desde qué punta a qué punto de la matriz están los datos, coordenadas **X** e **Y**, que tenemos que comprobar. En la línea siguiente, 110, miramos si el primer punto está coloreado con la rutina que empieza en la línea 200, y si no lo está, lo coloreamos con la instrucción **PLOT** y lo guardamos en la «cola» para comprobar más adelante los cuatro puntos contiguos. De la línea 120 a 160 lo que hacemos es sacar un dato de la cola, siempre el primero que haya esperando en ella, y comprobamos los cuatro puntos adyacentes de la misma forma que hicimos con el primero. La línea 170 sirve para comprobar si la «cola» está vacía; en ese caso, es que no queda ningún punto por comprobar y hemos llegado al final del rellenado, con lo que la rutina no salta a la línea 120 a comprobar el siguiente punto de la «cola», y finaliza la ejecución.

Hemos hecho este programa en forma de subrutina para poder utilizarlo en cualquier programa de gráficos y poder colorear cualquier figura.

Su uso es muy sencillo, simplemente escogemos un punto del interior de la figura y lo asignamos a X1 e Y1, y seguidamente llamamos a la rutina con GOSUB 100. Por ejemplo, para colorear la figura con forma de punta de flecha, hacemos lo siguiente:

```
   10 REM ************************
   20 REM * PROG. DE COLOREADO 2*
   30 REM ************************
   35 DIM p(500,2)
   37 DEF FN z(v)=INT ((((v+1)/50
0)-(INT ((v+1)/500)))*500)+1
   40 CLS : PLOT 100,50: DRAW 50,
```

```
-50: DRAW -50,100: DRAW -50,-100
: DRAW 50,50
  50 FOR Y=0 TO 175
  60 LET X1=0
  70 IF POINT (X1,Y1)=1 THEN
   GO TO 100
  80 LET X1=X1+1
  90 IF X1<=255 THEN GO TO 70
 100 LET pr=1: LET ul=0
 110 LET xa=x1: LET ya=y1: GO SU
B 200
 120 LET x=p(pr,1): LET y=p(pr,2
): LET pr=FN z(pr)
 130 LET xa=x: LET ya=y+1: GO SU
B 200
 140>LET xa=x: LET ya=y-1: GO SU
B 200
 150 LET xa=x+1: LET ya=y: GO SU
B 200
 160 LET xa=x-1: LET ya=y: GO SU
B 200
 170 IF pr<>FN z(ul) THEN GO TO
120
 180 RETURN
 200 IF POINT (xa,ya)>0 THEN RET
URN
 210 PLOT xa,ya
 220 LET ul=FN z(ul)
 230 LET p(ul,1)=xa: LET p(ul,2)
=ya
 240 RETURN
```

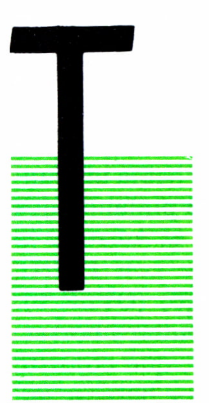
```
10 REM **************************
20 REM * PROG. DE COLOREADO 3   *
30 REM **************************
40 DIM P(500,2)
45 MODE 1:CLG
47 ORIGIN 0,0:PLOT 100,50:DRAWR 50,-50:
   DRAWR -50,100:DRAWR -50,-100:DRAWR 50,50
50 X1=101:Y1=70
60 GOSUB 100
70 END
100 PR=1:UL=0
110 XA=X1:YA=Y1:GOSUB 200
120 X=P(PR,1):Y=P(PR,2): pr=(pr+1) MOD 500
130 XA=X:YA=Y+2:GOSUB 200
140 XA=X:YA=Y-2:GOSUB 200
```

```
150 XA=X+2:YA=Y:GOSUB 200
160 XA=X-2:YA=Y:GOSUB 200
170 IF PR<>((ul+1) MOD 500) THEN GOTO 120
180 RETURN
200 IF TEST (XA,YA)>0 THEN RETURN
210 PLOT XA,YA
220 UL= (ul+1) MOD 500
230 P(UL,1)=XA:P(UL,2)=YA
240 RETURN
```

Esta rutina es bastante lenta, por lo que es recomendable usarla sólo para figuras de pequeño tamaño. Por contra, podemos tener la seguridad de que rellenará correctamente cualquier figura.

Puedes utilizarla con las figuras del capítulo anterior o inventar figuras irregulares y rellenarlas de varios colores cambiando el color de la tinta para cada figura. Por último, ten cuidado de que el punto donde te sitúes para empezar a colorear no esté ya pintado, y de que la figura sea realmente cerrada, ya que si fuera una figura abierta el programa intentaría rellenar toda la pantalla.

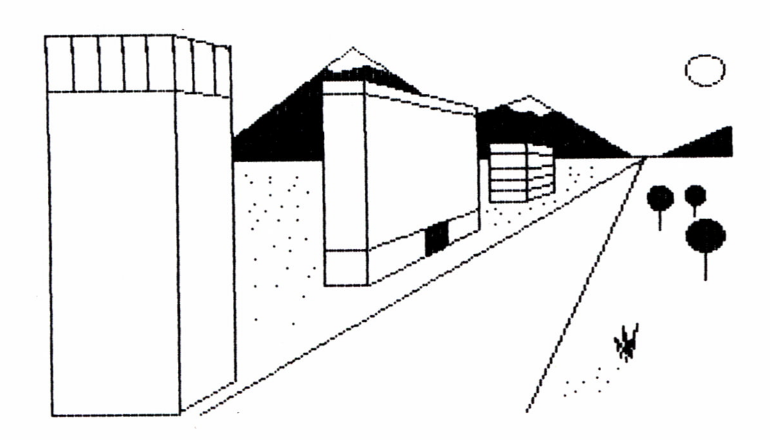
EL PINCEL DE BROCHA FINA 10



TAL y como hemos hecho con los gráficos de baja resolución, vamos ahora a aplicar todas las técnicas que hemos aprendido sobre el manejo de gráficos de alta resolución, para realizar un programa de dibujo en la pantalla del ordenador, es decir, vamos a volver a convertir a nuestro ordenador en un lienzo donde pintas un dibujo, en este caso con un pincel fino.

El programa va a consistir en un punto, nuestro pincel, que se va a ir moviendo por la pantalla y con el cual vamos a poder trazar líneas rectas, rectángulos y círculos. También vamos a poder colorear el interior de las figuras dibujadas y poner puntos simples en la pantalla. Por supuesto, podremos cambiar el color del pincel a nuestro gusto.

Un ejemplo de lo que podremos hacer es la siguiente figura.



PRINCIPIOS BASICOS DEL PROGRAMA

El concepto fundamental que vamos a aplicar en este programa es el mismo que el del capítulo 5, o sea, un bucle que se repite indefinidamente, en el cual leemos el teclado y en función de la tecla leída hacemos una cosa u otra. Si recordamos el programa del capítulo 5, esto se conseguía mediante la función INKEY$ que nos permitía leer directamente el teclado. Asimismo, tendremos una serie de teclas que nos permitirán movernos en la pantalla, más concretamente, I,J,K y M en el Spectrum, y los cursores en el Amstrad; por supuesto, las teclas se pueden cambiar a gusto del programador.

Otro concepto importante es la función OVER, que si está activada (OVER 1), al pintar un punto en la pantalla, si en ese sitio ya había un punto pintado, lo borra, y si no había nada, lo pinta. O sea, que pintando dos veces en el mismo sitio conseguimos borrar un punto. En el Amstrad la función OVER no existe, pero se simula perfectamente con PRINT CHR$(23); CHR$(1), (OVER 1) y PRINT CHR$(23);CHR$(0) (OVER 0). Esta función nos va a permitir utilizar la misma rutina para pintar y borrar.

También es importante el concepto de «transparencia», que consiste que al pintar un punto en la pantalla se conserva el color que hubiese antes, si había algún punto pintado en la pantalla. En el Spectrum esto se logra con la función INK 8, mientras que en el Amstrad el modo transparente se conecta con PRINT CHR$(22);CHR$(1) y se desconecta con PRINT CHR$(22);CHR$(0).

La elección de la tinta y el color del papel varían de un ordenador a otro, como es natural, ya que trabajan de forma diferente.



DESCRIPCION DEL PROGRAMA

Ahora vamos a estudiar el programa de una forma detallada, analizando cada parte por separado.

En la primera parte del programa, de las líneas 10 a la 60, inicializamos una serie de variables que nos van a hacer falta para el funcionamiento del programa. La matriz P es una matriz auxiliar que se utiliza en el proceso de rellenado. En el listado del Spectrum creamos una función con DEF FN que nos va a servir para simular la función MOD, que calcula el resto de la división, ya que el Spectrum carece de ella. También inicializamos el modo de pintar (OVER 1), y el color con el que vamos a pintar, almacenándolo en una variable auxiliar llamada COLOR. Borramos la pantalla e inicializamos las coordenadas x e y del punto a pintar a 0,0; al pun-

to inferior izquierdo de la pantalla. En la línea 50 inicializamos unas variables auxiliares L,C,R, a cero, que son las que nos van a decir en el programa si estamos pintando una línea (L=1); un círculo (C=1) o un rectángulo (R=1). Y en la línea 60 pintamos el pincel en las coordenadas x,y.

En las líneas 70 a 290 está contenido el bucle principal de control. Empieza con una sentencia para leer una tecla del teclado del ordenador. Las cuatro siguientes líneas controlan el movimiento del pincel en la pantalla. Las cuatro líneas funcionan igual. En la condición de las IF comprobamos que se ha pulsado la tecla correspondiente y que no estamos en los límites de la pantalla. Seguidamente saltamos a la subrutina en la línea 300, que es la que se encarga de pintar o borrar en la pantalla, en este caso borrar. Actualizamos más adelante la coordenada correspondiente a la tecla y volvemos a llamar a la subrutina en la línea 300, en este caso para volver a pintar en la pantalla.

Las líneas 120 a 140 se encargan de controlar las teclas L,C,R, que son las que nos van a permitir pintar una línea, un círculo o un rectángulo. Si pulsamos una de estas teclas, luego con las teclas de movimiento podremos variar el tamaño y dirección de la figura pintada sin modificar el dibujo que ya hubiera. Estas teclas sólo permiten coger un solo modo a la vez.

Las líneas 150 y 160 nos permiten fijar o borrar definitivamente lo que estamos dibujando provisionalmente en la pantalla. Por ejemplo, si estamos dibujando un rectángulo, al pulsar F el dibujo se «fija» en la pantalla, y si pulsamos B, el rectángulo desaparece de la pantalla, con lo que esta tecla es útil para borrar un dibujo previamente fijado. Si pulsamos F mientras nos movemos sin trazar ningún dibujo, fijamos un punto en la pantalla, y esto puede ser útil para dar efectos de sombreado a los dibujos. En el caso de la tecla F lo que hacemos es quitar el modo OVER 1 y pintar el dibujo, luego restablecemos las condiciones normales. Para la tecla B es igual, pero borrando en vez de pintar.

Las líneas 170 y 180 sirven para cambiar el color del pincel y el color del fondo. En el caso del color del pincel basta con pulsar un número dentro del rango de colores y automáticamente el color del trazo dejado cambiará. Al cambiar el color del papel además la pantalla se borrará, permitiéndonos comenzar un dibujo nuevo.

En la línea 190 detectamos la letra e, que nos va a servir para rellenar un área de un color determinado. Para ello, en la línea ponemos el color de la tinta, quitamos el modo OVER 1 y llamamos a la rutina de rellenar de color.

En la mayoría de las sentencias anteriores, en los IF aparece la condición (L+R+C)=0, esto es para evitar mezclar dibujos distintos o rellenar, o sea, mientras estamos pintando algo no podemos hacer otra cosa. También en esas líneas aparecen 2 variables X_1 e Y_1, que son las coordenadas donde hemos empezado a pintar algo. Es como si el punto X_1, Y_1 lo hubiéramos pinchado con un alfiler y a partir de ahí empezamos a pintar algo.

En la línea 300 empieza la subrutina de pintar, y lo que hace es comprobar si hay algún modo puesto (C=1, L=1 o R=1) y pintar una cosa u otra en cada caso.

En el caso de L=1 pinta una línea de X_1, Y_1 a X, Y. Si R=1, pinta un rectángulo con dos vértices opuestos en X_1, Y_1 y X, Y.

Si es C=1, pinta una circunferencia de centro X_1, Y_1 y de radio hasta X, Y.

Si no hay ningún tipo de dibujo activado, simplemente pinta un punto.

En la línea 400 empieza la rutina de coloreado, tomando como punto inicial el punto X_1, Y_1. Esta rutina colorea cualquier tipo de figura, pero es muy lenta, con lo que sólo es útil para colorear regiones delimitadas por líneas pequeñas. No es conveniente colorear una figura abierta, ya que no detecta los bordes de la pantalla y el programa se pararía dando un error.

Vamos a hacer un resumen de las teclas disponibles en este programa.

I ˆ[“ˆ\ : movimiento hacia arriba
M ˆ[“ˆ\ : movimiento hacia abajo
J ˆ[“ˆ\ : movimiento a la izquierda
K ˆ[“ˆ\ : movimiento a la derecha
L : empieza el modo línea. Cualquier desplazamiento provocará un aumento o disminución de la línea
R : Empieza el modo rectángulo. Cualquier desplazamiento provocará un aumento o disminución del tamaño del rectángulo.
C : Empieza el modo círculo. Con las teclas de desplazamiento conseguiremos el aumento o disminución del tamaño.
F : «Fija» el dibujo en la pantalla y termina cualquier modo activo.
B : Borra el dibujo de la pantalla y termina el modo activo.
0 : (número de color), pone el pincel de un color determinado.
V : Cambia el color del fondo y borra la pantalla.
E : Rellena un contorno cerrado del color del pincel partiendo de la posición actual.

Evidentemente se pueden añadir muchas más opciones, pero las más interesantes son:

— Posibilidad de guardar y coger de la cinta o disco un dibujo ya hecho. En el Spectrum esto se puede realizar con una sentencia tal como la siguiente:

```
200 IF T$ = "G" THEN SAVE "FIGURA" SCREEN$
```

y en el Amstrad:

```
200 IF T$ = "G" THEN SAVE "FIGURA",b,&C000,&3FCF
```

Para más información, mirar los manuales correspondientes.

— Tecla para finalizar el programa.

Los sistemas de dibujo de este programa son los mismos, muy simplificados naturalmente, que los que emplean los sistemas de Diseño Asistido por Ordenador, CAD.

Las posibilidades del programa y de la alta resolución no se han acabado aquí, hay un montón de técnicas que no hemos explicado, pero con los conocimientos adquiridos en este libro se podrá investigar y descubrir una gran cantidad de ellas. Animo y a programar.

```
10 REM ******************************
20 REM *    PROG. DE DIBUJO FINO    *
30 REM *       VERSION AMSTRAD      *
33 REM ******************************
35 DIM P(500,2):INK 0,0
40 MODE 1:PRINT CHR$(23);CHR$(1);:
   CO=1:Y=0:X=0:CLS:CLG:PEN 1
50 L=0:C=0:R=0
60 PLOT X,Y,CO
70 T$=INKEY$:IF T$="" THEN GOTO 70
80 IF T$=CHR$(240) AND Y<399 THEN CP=CO:
   GOSUB 300:Y=Y+2:GOSUB 300
90 IF T$=CHR$(241) AND Y>0 THEN CP=CO:
   GOSUB 300:Y=Y-2:GOSUB 300
100 IF T$=CHR$(242) AND X>0 THEN CP=CO:
    GOSUB 300:X=X-2:GOSUB 300
110 IF T$=CHR$(243) AND X<638 THEN CP=CO:
    GOSUB 300:X=X+2:GOSUB 300
120 IF T$="L" AND (L+R+C)=0  THEN L=1:X1=X:Y1=Y
130 IF T$="C" AND (L+R+C)=0  THEN C=1:X1=X:Y1=Y
140 IF T$="R" AND (L+R+C)=0  THEN R=1:X1=X:Y1=Y
150 IF T$="F" THEN PRINT CHR$(23);CHR$(0);:CP=CO:
GOSUB 300:PRINT CHR$(23);CHR$(1);:PLOT X,Y,CO:L=0:C=0
:R=0
160 IF T$="B" THEN PRINT CHR$(23);CHR$(0);:CP=0:
GOSUB 300:PRINT CHR$(23);CHR$(1);:PLOT X,Y,CO:L=0:C=0
:R=0
170 IF T$>="0" AND T$<="3" THEN CP=CO:GOSUB 300:
CO=VAL(T$):CP=CO:GOSUB 300
```

```
180 IF T$="V" AND (L+R+C)=0 THEN LOCATE 1,25:
INPUT "COLOR DEL FONDO :";A:IF A>=0 AND A<=26 THEN I
NK 0,A:CLS:CLG:X=0:Y=0:PLOT X,Y
190 IF T$="E" AND (L+R+C)=0 THEN PRINT CHR$(23);CHR$(0);
:PLOT X,Y,0:X1=X:Y1=Y:GOSUB 400:X=X1:Y=Y1:PRINT CHR$
(23);CHR$(1);:PLOT X,Y,CO
290 GOTO 70
300 REM * SUBRUTINA DE PINTAR *
310 IF L=1 THEN PLOT X1,Y1,CP : DRAWR X-X1,Y-Y1,CP
320 IF C=1 THEN ORIGIN X1,Y1:GOSUB 600:ORIGIN 0,0
330 IF (L+R+C)=0 THEN PLOT X,Y,CP
340 IF R=1 THEN PLOT X1,Y1,CP:DRAWR X-X1,0,CP:
DRAWR 0,Y-Y1,CP:DRAWR X1-X,0,CP: DRAWR 0,Y1-Y,CP
350 REM * FIN DE LA SUBRUTINA *
390 RETURN
400 PR=1 : UL=0
410 XA=X1:YA=Y1:GOSUB 500
420 X=P(PR,1):Y=P(PR,2):PR=(PR+1) MOD 500
430 XA=X:YA=Y+2:GOSUB 500
440 XA=X:YA=Y-2:GOSUB 500
450 XA=X+2:YA=Y:GOSUB 500
460 XA=X-2:YA=Y:GOSUB 500
470 IF PR<>((UL+1) MOD 500) THEN GOTO 420
480 RETURN
500 IF TEST(XA,YA)>0 THEN RETURN
510 PLOT XA,YA,CO
520 UL=(UL+1) MOD 500
530 P(UL,1)=XA:P(UL,2)=YA
540 RETURN
600 R=SQR(ABS(X1-X)^2+ABS(Y1-Y)^2)
610 MOVE 0,R:CA=COS(PI/90):SA=SIN(PI/90)
620 X2=0:Y2=R
630 FOR I=0 TO 180
640 X3=X2*CA-Y2*SA:Y3=Y2*CA+X2*SA:DRAW X3,Y3,CP
650 X2=X3:Y2=Y3:NEXT I
660 RETURN
```

```
  10 REM ************************
  20 REM * prg. de Dibujo fino *
  30 REM ************************
  33 DIM p(500,2)
  35 DEF FN z(v)=INT ((((v+1)/50
0)-(INT ((v+1)/500)))*500)+1
  40 OVER 1: LET color=0: INK 0:
 LET y=0: LET x=0: CLS
  50 LET l=0: LET c=0: LET r=0
  60 PLOT x,y
```

```
  70 LET t$=INKEY$: IF t$="" THE
N GO TO 70
  80 IF t$="i" AND y<175 THEN GO
 SUB 300: LET y=y+1: GO SUB 300
  90 IF t$="m" AND y>0 THEN GO S
UB 300: LET y=y-1: GO SUB 300
 100 IF t$="j" AND x>0 THEN GO S
UB 300: LET x=x-1: GO SUB 300
 110 IF t$="k" AND x<255 THEN GO
 SUB 300: LET x=x+1: GO SUB 300
 120 IF t$="l" AND (l+r+c)=0 THE
N LET l=1: LET x1=x: LET y1=y
 130 IF t$="c" AND (l+r+c)=0 THE
N LET c=1: LET x1=x: LET y1=y
 140 IF t$="r" AND (l+r+c)=0 THE
N LET r=1: LET x1=x: LET y1=y
 150 IF t$="f" THEN INK COLOR: O
VER 0: GO SUB 300: OVER 1: PLOT
x,y: LET l=0: LET c=0: LET r=0:
INK 8
 160 IF t$="b" THEN INK COLOR: O
VER 0: INVERSE 1: GO SUB 300: IN
VERSE 0: OVER 1: PLOT x,y: LET l
=0: LET c=0: LET r=0: INK 8
 170 IF t$>="0" AND t$<="7" THEN
 GO SUB 300: LET color=VAL t$: G
O SUB 300
 180 IF t$="v" AND (l+r+c)=0 THE
N INPUT "color del fondo ";a$: I
F a$(1)>="0" AND a$(1)<="7" THEN
 PAPER VAL a$(1): CLS : LET x=0:
 LET y=0
 190 IF t$="e" AND (l+r+c)=0 THE
N PLOT x,y: OVER 0: INK color: L
ET x1=x: LET y1=y: GO SUB 400: L
ET x=x1: LET y=y1: OVER 1: PLOT
x,y: INK 8
 290 GO TO 70
 300 REM * subrutina de pintar *
 310 IF l=1 THEN PLOT x1,y1: DRA
W x-x1,y-y1
 320 IF c=1 THEN CIRCLE x1,y1,SQ
R (ABS (x1-x)↑2+ABS (y1-y)↑2)
 330 IF (l+r+c)=0 THEN INK 8: PL
OT x,y
 340 IF r=1 THEN PLOT x1,y1: DRA
W x-x1,0: DRAW 0,y-y1: DRAW x1-x
,0: DRAW 0,y1-y

 350 REM * fin de la subrutina *
```

```
 390 RETURN
 400 LET pr=1: LET ul=0
 410 LET xa=x1: LET ya=y1: GO SU
B 500
 420 LET x=p(pr,,1): LET y=p(pr,2
): LET pr=FN z(pr)
 430 LET xa=x: LET ya=y+1: GO SU
B 500
 440 LET xa=x: LET ya=y-1: GO SU
B 500
 450 LET xa=x+1: LET ya=y: GO SU
B 500
 460 LET xa=x-1: LET ya=y: GO SU
B 500
 470 IF pr<>FN z(ul) THEN GO TO
420
 480 RETURN
 500 IF POINT (xa,ya)>0 THEN RET
URN
 510 PLOT xa,ya
 520 LET ul=FN z(ul)
 530 LET p(ul,1)=xa: LET p(ul,2)
=ya
 540 RETURN
```

APENDICES

APENDICE SOBRE GRAFICOS EN IBM

Los ordenadores IBM disponen de una gran potencia gráfica que nos permite resolver prácticamente cualquier problema sobre dibujo que podamos plantearnos.

En el presente apéndice vamos a hacer un breve repaso de los gráficos en baja resolución, así como un estudio resumido de las principales sentencias gráficas para alta resolución que podemos utilizar en un IBM. Finalmente, veremos algunos programas de ejemplo.



Baja resolución

La pantalla de baja resolución (o pantalla de texto) es la que presenta el ordenador nada más encenderlo. Dispone de 25 filas y 80 columnas (numeradas de 1 a 25 y de 1 a 80, respectivamente). Sin embargo, también podemos disponer de una pantalla de 40 columnas.

Para cambiar de un ancho de pantalla al otro existe el comando: WIDTH A donde A valdrá 40 u 80, según el número de columnas que deseemos. Por otra parte, si por cualquier motivo nos encontráramos en una pantalla gráfica (alta resolución) y quisiéramos pasar a la pantalla de texto, el comando SCREEN 0 nos resuelve este problema.

Para imprimir en una posición determinada de la pantalla de texto tenemos que teclear, antes de la instrucción PRINT correspondiente, la sentencia LOCATE X,Y donde X indica el número de fila e Y el número de columna. El origen de filas y columnas se encuentra en el ángulo superior izquierdo (posición 1.1). Por otra parte, aunque la pantalla cuenta con 25 líneas conviene no imprimir por debajo de la línea 22, ya que el IBM reserva las tres líneas inferiores para mensajes de error o conformidad (OK), introducción de nuevas órdenes e impresión en pantalla de claves de funciones (F1 ...F10).

El IBM cuenta, además de los caracteres estándar (letras, números y signos), con una gran variedad de caracteres gráficos, cada uno de los cuales tiene asignado un número del código ASCII. Por tanto, para imprimir uno de estos caracteres no tenemos más que teclear: PRINT CHR$(N), donde N es un número entre 0 y 255 que representa el código ASCII asociado al carácter que deseamos imprimir.

En cuanto a los colores, el IBM puede cambiar los colores de la tinta, el fondo (papel) y el borde con una sola sentencia: COLOR, que tiene el siguiente formato:

COLOR [T][,[F][,B]]

Donde T es un número entre 0 y 31 que representa el color de la tinta, F es un número entre 0 y 7 para el color del fondo, y B es un número entre 0 y 15 que define el color para el borde de la pantalla.

Los colores disponibles son los siguientes:

0	Negro	8	Gris
1	Azul	9	Azul claro
2	Verde	10	Verde claro
3	Azul ultramar	11	Azul ultramar claro
4	Rojo	12	Rojo claro
5	Magenta	13	Magenta claro
6	Marrón	14	Amarillo
7	Blanco	15	Blanco intenso

Podemos observar que los parámetros del 16 al 31, admisibles para la tinta, no están incluidos en esta tabla. Esto es debido a que vuelve a ser la misma secuencia de colores, pero en estado intermitente, es decir, los caracteres aparecen en pantalla parpadeando.

Alta resolución

La pantalla de alta resolución (o pantalla gráfica) del IBM puede alcanzar dos niveles distintos de resolución: 320 por 200 puntos o 640 por 200 puntos. Ya vimos en el apartado anterior que SCREEN 0 nos proporcionaba la pantalla de texto; pues bien, el comando SCREEN 1 nos proporciona la pantalla gráfica de media resolución (320 por 200) y el comando SCREEN 2 da paso a la pantalla gráfica de alta resolución (640 por 200). SCREEN puede usarse también como instrucción dentro de una línea de programa. En ambas pantallas gráficas el origen de coordenadas está en el ángulo superior izquierdo, en el punto (0,0).



Principales sentencias gráficas

Nota aclaratoria: Los parámetros entre corchetes son opcionales, por tanto, se pueden suprimir, y el ordenador tomará por defecto el valor que esté definido en el momento de la ejecución.



Pset y Preset

```
PSET (x,y)[,color]
```

Dibuja un punto en la posición de la pantalla especificada por las coordenadas (x,y).

```
PRESET (x,y)[,color]
```

Tiene la misma misión que PSET con la única diferencia de que, si no se especifica el parámetro *color*, selecciona el mismo color del fondo para dibujar el punto de coordenadas (x,y).

Ejemplo:

```
10 SCREEN 1
20 PSET (160,100)
30 FOR I=1 TO 5000:NEXT
40 PRESET (160,100)
```

El programa dibuja un punto en el centro de la pantalla gráfica de media resolución y tras una breve pausa la borra (imprime un punto con el color del fondo).



Line

```
LINE [(X1,Y1)]-(X2,Y2)[,[color][,B[F]]][,estilo]]
```

Dibuja una línea o un cuadrado en la pantalla. La forma más sencilla: LINE [(X1,Y1)]-(X2,Y2) traza una línea en la pantalla desde el punto de coordenadas (X1,Y1) hasta el punto (X2,Y2). Si no especificamos el primer punto, la línea se traza desde el punto donde estuviera posicionado el cursor gráfico en ese momento.

El argumento *B* produce el dibujo de un rectángulo con los puntos (X1,Y1) y (X2,Y2) como vértices opuestos.

El argumento *BF* también dibuja un rectángulo y además lo rellena con el color seleccionado.

La opción *estilo* sirve para trazar líneas discontinuas o punteadas. Este último parámetro se introduce en forma de número hexadecimal y representa una secuencia de 16 bits con 0 ó 1. Los 1 se traducen en puntos encendidos en la pantalla y los 0 serán puntos apagados.

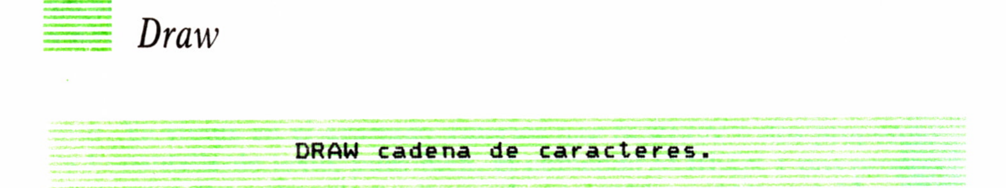
La opción *estilo* no puede incluirse junto a BF, ya que no tiene sentido dibujar un rectángulo con las líneas punteadas y rellenarlo de color. Ejemplo:

```
10 SCREEN 1
20 LINE (100,40)-(220,160), , BF
30 LINE (100,170)-(220,170), , , &HAAA
```

El programa dibuja un cuadrado en el centro de la pantalla de media resolución y lo rellena de color (línea 20). Como no está especificado el color, utiliza el que está definido para la tinta en el momento de la ejecución.

La línea 30 traza una línea de puntos debajo del cuadrado.

Draw



Traza un dibujo según se especifique en la *cadena de caracteres*. DRAW utiliza un *lenguaje de definición de gráficos*. Los mandatos de dicho lenguaje son los que deben incluirse en la expresión *cadena de caracteres*.

Nota: el parámetro *n* indica la distancia a desplazar.
Un movimiento hacia arriba
Dn movimiento hacia abajo
Ln movimiento hacia la izquierda
Rn movimiento hacia la derecha
En movimiento en diagonal hacia arriba y derecha
Fn movimiento en diagonal hacia abajo y derecha
Gn movimiento en diagonal hacia abajo e izquierda
Hn movimiento en diagonal hacia arriba e izquierda

Los dos mandatos siguientes pueden preceder a cualquiera de los anteriores:

B movimiento sin trazado de puntos
N movimiento y retorno a la posición original cuando termine
P color, límite
permite colorear figuras. El parámetro *color* es el color para la figura y el *límite* es el color del borde.

Existen algunos mandatos más para la sentencia DRAW que podemos encontrar en el Manual BASIC de IBM.

Ejemplo:

```
10 SCREEN 1
20 PRESET (110,150)
30 DRAW "U100R100D100L100"
40 DRAW "BE50"
50 DRAW "P1,3"
```

El programa sitúa el cursor gráfico en el punto (110,150) (línea 20). En la línea 30 dibuja un cuadrado. En la línea 40 el cursor gráfico se mueve en diagonal (arriba y derecha) hacia dentro del cuadro. Por último, en la línea 50 pinta el interior del cuadrado.



Circle

```
CIRCLE (x,y), r [,color [comienzo,
final [,aspecto ]]]
```

Dibuja elipses o circunferencias en la pantalla con centro en el punto de coordenadas (x,y) y radio (eje principal de la elipse) *r*.

Los parámetros *comienzo* y *final* son ángulos en radianes y pueden variar entre –2*PI y 2*PI. Determinan dónde comienza y termina el trazado de la elipse, permitiendo así trazar arcos.

El parámetro *aspecto* es un número que da la relación entre los dos ejes principales de la elipse. Si se omite el resultado será una circunferencia.

Ejemplo:

```
10 SCREEN 1
20 CIRCLE (160,100),60,,,,5/18
```

El programa traza una elipse de eje mayor (horizontal) igual a 60. Si el aspecto fuera mayor que 1 el eje mayor sería el vertical.



Color

```
COLOR [fondo][,paleta]
```

Selecciona el color del papel (fondo de la pantalla). El parámetro *fondo* es un número entre 0 y 15 que especifica el color del fondo. El parámetro *paleta* toma valor 0 ó 1 y selecciona una de las dos paletas disponibles para el color de la tinta.

Color	Paleta 0	Paleta 1
1	Verde	Cyan
2	Rojo	Magenta
3	Marrón	Blanco
0	Color del fondo	Color del fondo

Por tanto, los parámetros *color* vistos en las anteriores sentencias sólo pueden tomar valores comprendidos entre 0 y 3 en media resolución (SCREEN 1). En alta resolución (SCREEN 2) sólo hay dos colores disponibles para la tinta: el mismo color del fondo (0) y el blanco (1). Los valores tomados por defecto son 0 (negro) para el fondo y 3 para la tinta (marrón en la paleta 0 y blanco en la 1).

Ejemplo:

```
10 SCREEN 1
20 COLOR 1,0
30 LINE (50,50)-(270,150),2,BF
```

El programa selecciona en la línea 20 el color 1 (azul) para el fondo y la paleta 0. En la línea 30 traza un rectángulo y lo colorea con el color 2 (rojo en la paleta 0).



Paint

```
PAINT (x,y) [[,color] [,límite][,fondo]]
```

Rellena un área de la pantalla con el color seleccionado. Las coordenadas (x,y) corresponden a un punto dentro del área que va a rellenarse. El parámetro *color,* igual que en las demás sentencias, indica el color con el que se va rellenar la figura y, por tanto, podrá tomar un valor entre 0 y 3 en media resolución y los valores 0 ó 1 en alta resolución. El parámetro *límite* determina el color de los bordes de la figura a rellenar, por tanto, tiene las mismas características que *color*. Por último, *fondo* es un carácter gráfico definido por el usuario para pintar.

Ejemplo:

```
10 SCREEN 1
20 CIRCLE (160,100),80
30 PAINT (160,100),2,3
```

El programa traza una circunferencia de radio 80 en el centro de la pantalla (línea 20) y a continuación la rellena con el color 2 (rojo o magenta dependiendo de la paleta que esté seleccionada en el momento de la ejecución). El límite es de color 3 (marrón o blanco), ya que al no estar especificado en la sentencia CIRCLE toma el valor máximo de la gama.



Step

Esta sentencia no es propiamente de tipo gráfico, pero se utiliza ligada a ellos. Sirve para dar las coordenadas (x,y) en forma relativa en lugar de en forma absoluta que es como lo hemos visto hasta ahora.

Ejemplo:

```
10 SCREEN 1
20 CIRCLE (160,100),80
30 PAINT STEP (0,0),2,3,
```

Este programa tiene el mismo objetivo que el anterior, pero utiliza coordenadas relativas en la sentencia **PAINT** y están referidas al último punto posicionado que es el centro de la circunferencia y de la pantalla (160,100) en coordenadas absolutas.

Hasta aquí hemos hecho una breve descripción de las principales sentencias gráficas. Existen algunas más que nos permiten todo tipo de «sofisticaciones» gráficas y que podemos encontrarlas explicadas detalladamente en el Manual de BASIC de IBM.

Por otra parte, las sentencias gráficas más potentes sólo existen en las versiones de BASIC Avanzado, por lo que conviene utilizar este tipo de BASIC si queremos evitar mensajes de error.



CODIGO ASCII DEL AMSTRAD

CARACTER	CHR$	CARACTER	CHR$	CARACTER	CHR$	CARACTER	CHR$
	0		32	@	64	`	96
	1	!	33	A	65	a	97
	2	"	34	B	66	b	98
	3	#	35	C	67	c	99
	4	$	36	D	68	d	100
	5	%	37	E	69	e	101
✓	6	&	38	F	70	f	102
Ω	7	'	39	G	71	g	103
←	8	(	40	H	72	h	104
→	9	)	41	I	73	i	105
↓	10	*	42	J	74	j	106
↑	11	+	43	K	75	k	107
	12	,	44	L	76	l	108
	13	-	45	M	77	m	109
	14	.	46	N	78	n	110
	15	/	47	O	79	o	111
	16	0	48	P	80	p	112
	17	1	49	Q	81	q	113
	18	2	50	R	82	r	114
	19	3	51	S	83	s	115
	20	4	52	T	84	t	116
	21	5	53	U	85	u	117
	22	6	54	V	86	v	118
	23	7	55	W	87	w	119
	24	8	56	X	88	x	120
	25	9	57	Y	89	y	121
	26	:	58	Z	90	z	122
	27	;	59	[	91	{	123
	28	<	60	\	92	|	124
	29	=	61	]	93	}	125
	30	>	62	↑	94	~	126
	31	?	63	_	95		127

CARACTER	CHR$	CARACTER	CHR$	CARACTER	CHR$	CARACTER	CHR$
	128		160		192		224
	129		161		193		225
	130		162		194		226
	131		163		195		227
	132		164		196		228
	133		165		197		229
	134		166		198		230
	135		167		199		231
	136		168		200		232
	137		169		201		233
	138		170		202		234
	139		171		203		235
	140		172		204		236
	141		173		205		237
	142		174		206		238
	143		175		207		239
	144		176		208		240
	145		177		209		241
	146		178		210		242
	147		179		211		243
	148		180		212		244
	149		181		213		245
	150		182		214		246
	151		183		215		247
	152		184		216		248
	153		185		217		249
	154		186		218		250
	155		187		219		251
	156		188		220		252
	157		189		221		253
	158		190		222		254
	159		191		223		255



CODIGO ASCII DEL COMMODORE

	0	CRSR ↓	17	"	34	3	51
	1	RVS ON	18	#	35	4	52
	2	CLR HOME	19	$	36	5	53
	3	INST DEL	20	%	37	6	54
	4		21	&	38	7	55
WHT	5		22	'	39	8	56
	6		23	(	40	9	57
	7		24	)	41	:	58
ANULAR SHIFT C=	8		25	*	42	;	59
PONER SHIFT C=	9		26	+	43	<	60
	10		27	,	44	=	61
	11	RED	28	−	45	>	62
	12	CRSR →	29	.	46	?	63
RETURN	13	GRN	30	/	47	@	64
CONMUTADOR MINUSCULAS	14	BLU	31	0	48	A	65
	15	SPACE	32	1	49	B	66
	16	!	33	2	50	C	67

D	68		97		126		155
E	69		98		127	PUR	156
F	70		99		128	CRSR ←	157
G	71		100		129	YEL	158
H	72		101		130	CYN	159
I	73		102		131	SPACE	160
J	74		103		132		161
K	75		104	f1	133		162
L	76		105	f3	134		163

M	77		106	f5	135		164
N	78		107	f7	136		165
O	79		108	f2	137		166
P	80		109	f4	138		167
Q	81		110	f6	139		168
R	82		111	f8	140		169
S	83		112	SHIFT RETURN	141		170
T	84		113	CONMUTADOR MAYUSCULAS	142		171
U	85		114		143		172
V	86		115	BLK	144		173
W	87		116	CRSR	145		174
X	88		117	RVS OFF	146		175
Y	89		118	CLR HOME	147		176
Z	90		119	INST DEL	148		177
[	91		120		149		178
£	92		121		150		179
]	93		122		151		180
↑	94		123		152		181
←	95		124		153		182
	96		125		154		183

	184		186		188		190
	185		187		189		191



CODIGO ASCII DEL SPECTRUM

32	–		76	–	L	120	–	x
33	–	!	77	–	M	121	–	y
34	–	"	78	–	N	122	–	z
35	–	#	79	–	O	123	–	{
36	–	$	80	–	P	124	–	|
37	–	%	81	–	Q	125	–	}
38	–	&	82	–	R	126	–	~
39	–	'	83	–	S	127	–	©
40	–	(	84	–	T	128	–	
41	–	)	85	–	U	129	–	▝
42	–	*	86	–	V	130	–	▘
43	–	+	87	–	W	131	–	▀
44	–	,	88	–	X	132	–	▗
45	–	-	89	–	Y	133	–	▐
46	–	.	90	–	Z	134	–	▚
47	–	/	91	–	[	135	–	▜
48	–	0	92	–	\	136	–	▖
49	–	1	93	–	]	137	–	▞
50	–	2	94	–	↑	138	–	▌
51	–	3	95	–	_	139	–	▛
52	–	4	96	–	£	140	–	▄
53	–	5	97	–	a	141	–	▟
54	–	6	98	–	b	142	–	▙
55	–	7	99	–	c	143	–	█
56	–	8	100	–	d	144	–	A
57	–	9	101	–	e	145	–	B
58	–	:	102	–	f	146	–	C
59	–	;	103	–	g	147	–	D
60	–	<	104	–	h	148	–	E
61	–	=	105	–	i	149	–	F
62	–	>	106	–	j	150	–	G
63	–	?	107	–	k	151	–	H
64	–	@	108	–	l	152	–	I
65	–	A	109	–	m	153	–	J
66	–	B	110	–	n	154	–	K
67	–	C	111	–	o	155	–	L
68	–	D	112	–	p	156	–	M
69	–	E	113	–	q	157	–	N
70	–	F	114	–	r	158	–	O
71	–	G	115	–	s	159	–	P
72	–	H	116	–	t	160	–	Q
73	–	I	117	–	u	161	–	R
74	–	J	118	–	v	162	–	S
75	–	K	119	–	w	163	–	T

```
164 - U
165 - RND
166 - INKEY$
167 - PI
168 - FN
169 - POINT
170 - SCREEN$
171 - ATTR
172 - AT
173 - TAB
174 - VAL$
175 - CODE
176 - VAL
177 - LEN
178 - SIN
179 - COS
180 - TAN
181 - ASN
182 - ACS
183 - ATN
184 - LN
185 - EXP
186 - INT
187 - SQR
188 - SGN
189 - ABS
190 - PEEK
191 - IN
192 - USR
193 - STR$
194 - CHR$
195 - NOT
196 - BIN
197 - OR
198 - AND
199 - <=
200 - >=
201 - <>
202 - LINE
203 - THEN
204 - TO
205 - STEP
206 - DEF FN
207 - CAT
208 - FORMAT
209 - MOVE
210 - ERASE
211 - OPEN #
212 - CLOSE #
213 - MERGE
214 - VERIFY
215 - BEEP
216 - CIRCLE
217 - INK
218 - PAPER
219 - FLASH
220 - BRIGHT
221 - INVERSE
222 - OVER
223 - OUT
224 - LPRINT
225 - LLIST
226 - STOP
227 - READ
228 - DATA
229 - RESTORE
230 - NEW
231 - BORDER
232 - CONTINUE
233 - DIM
234 - REM
235 - FOR
236 - GO TO
237 - GO SUB
238 - INPUT
239 - LOAD
240 - LIST
241 - LET
242 - PAUSE
243 - NEXT
244 - POKE
245 - PRINT
246 - PLOT
247 - RUN
248 - SAVE
249 - RANDOMIZE
250 - IF
251 - CLS
252 - DRAW
253 - CLEAR
254 - RETURN
255 - COPY
```



TABLA DE CONVERSION DECIMAL-BINARIO

0 = 00000000	46 = 00101110	92 = 01011100
1 = 00000001	47 = 00101111	93 = 01011101
2 = 00000010	48 = 00110000	94 = 01011110
3 = 00000011	49 = 00110001	95 = 01011111
4 = 00000100	50 = 00110010	96 = 01100000
5 = 00000101	51 = 00110011	97 = 01100001
6 = 00000110	52 = 00110100	98 = 01100010
7 = 00000111	53 = 00110101	99 = 01100011
8 = 00001000	54 = 00110110	100 = 01100100
9 = 00001001	55 = 00110111	101 = 01100101
10 = 00001010	56 = 00111000	102 = 01100110
11 = 00001011	57 = 00111001	103 = 01100111
12 = 00001100	58 = 00111010	104 = 01101000
13 = 00001101	59 = 00111011	105 = 01101001
14 = 00001110	60 = 00111100	106 = 01101010
15 = 00001111	61 = 00111101	107 = 01101011
16 = 00010000	62 = 00111110	108 = 01101100
17 = 00010001	63 = 00111111	109 = 01101101
18 = 00010010	64 = 01000000	110 = 01101110
19 = 00010011	65 = 01000001	111 = 01101111
20 = 00010100	66 = 01000010	112 = 01110000
21 = 00010101	67 = 01000011	113 = 01110001
22 = 00010110	68 = 01000100	114 = 01110010
23 = 00010111	69 = 01000101	115 = 01110011
24 = 00011000	70 = 01000110	116 = 01110100
25 = 00011001	71 = 01000111	117 = 01110101
26 = 00011010	72 = 01001000	118 = 01110110
27 = 00011011	73 = 01001001	119 = 01110111
28 = 00011100	74 = 01001010	120 = 01111000
29 = 00011101	75 = 01001011	121 = 01111001
30 = 00011110	76 = 01001100	122 = 01111010
31 = 00011111	77 = 01001101	123 = 01111011
32 = 00100000	78 = 01001110	124 = 01111100
33 = 00100001	79 = 01001111	125 = 01111101
34 = 00100010	80 = 01010000	126 = 01111110
35 = 00100011	81 = 01010001	127 = 01111111
36 = 00100100	82 = 01010010	128 = 10000000
37 = 00100101	83 = 01010011	129 = 10000001
38 = 00100110	84 = 01010100	130 = 10000010
39 = 00100111	85 = 01010101	131 = 10000011
40 = 00101000	86 = 01010110	132 = 10000100
41 = 00101001	87 = 01010111	133 = 10000101
42 = 00101010	88 = 01011000	134 = 10000110
43 = 00101011	89 = 01011001	135 = 10000111
44 = 00101100	90 = 01011010	136 = 10001000
45 = 00101101	91 = 01011011	137 = 10001001

138 = 10001010
139 = 10001011
140 = 10001100
141 = 10001101
142 = 10001110
143 = 10001111
144 = 10010000
145 = 10010001
146 = 10010010
147 = 10010011
148 = 10010100
149 = 10010101
150 = 10010110
151 = 10010111
152 = 10011000
153 = 10011001
154 = 10011010
155 = 10011011
156 = 10011100
157 = 10011101
158 = 10011110
159 = 10011111
160 = 10100000
161 = 10100001
162 = 10100010
163 = 10100011
164 = 10100100
165 = 10100101
166 = 10100110
167 = 10100111
168 = 10101000
169 = 10101001
170 = 10101010
171 = 10101011
172 = 10101100
173 = 10101101
174 = 10101110
175 = 10101111
176 = 10110000
177 = 10110001
178 = 10110010
179 = 10110011
180 = 10110100
181 = 10110101
182 = 10110110
183 = 10110111
184 = 10111000
185 = 10111001
186 = 10111010
187 = 10111011
188 = 10111100
189 = 10111101
190 = 10111110
191 = 10111111
192 = 11000000
193 = 11000001
194 = 11000010
195 = 11000011
196 = 11000100
197 = 11000101
198 = 11000110
199 = 11000111
200 = 11001000
201 = 11001001
202 = 11001010
203 = 11001011
204 = 11001100
205 = 11001101
206 = 11001110
207 = 11001111
208 = 11010000
209 = 11010001
210 = 11010010
211 = 11010011
212 = 11010100
213 = 11010101
214 = 11010110
215 = 11010111
216 = 11011000
217 = 11011001
218 = 11011010
219 = 11011011
220 = 11011100
221 = 11011101
222 = 11011110
223 = 11011111
224 = 11100000
225 = 11100001
226 = 11100010
227 = 11100011
228 = 11100100
229 = 11100101
230 = 11100110
231 = 11100111
232 = 11101000
233 = 11101001
234 = 11101010
235 = 11101011
236 = 11101100
237 = 11101101
238 = 11101110
239 = 11101111
240 = 11110000
241 = 11110001
242 = 11110010
243 = 11110011
244 = 11110100
245 = 11110101
246 = 11110110
247 = 11110111
248 = 11111000
249 = 11111001
250 = 11111010
251 = 11111011
252 = 11111100
253 = 11111101
254 = 11111110
255 = 11111111

De entre las posibilidades que ofrecen los ordenadores una de las más sencillas, divertidas y apasionantes es, sin duda, la creación de gráficos.

Desde el primer «brochazo» aprenderá a diseñar y colorear tanto figuras sencillas como las más sofisticadas creaciones que puede llegar a imaginar, sin necesidad de profundos conocimientos informáticos ni artísticos.

Plasme sus ideas en atractivos programas llenos de color.